INTERNET-DRAFT
draft-hilland-rddp-verbs-00.txt

Jeff Hilland
  Hewlett-Packard Company
Paul Culley
  Hewlett-Packard Company
Jim Pinkerton
  Microsoft Corporation
Renato Recio
  IBM Corporation

Expires: October, 2003

RDMA Protocol Verbs Specification

1 Status of this Memo

2 Abstract

This document describes an abstract interface to a RDMA enabled NIC (RNIC). This interface is implemented as a combination of the RNIC, its associated firmware, and host software. It provides access to the RNIC queuing and memory management resources, as well as the underlying networking layers.

Table of Contents

Table of Figures

3 Introduction

This document describes an abstract interface to an RDMA aware NIC (RNIC). The RNIC implements the RDMA Protocol [RDMAP][DDP] above a reliable transport, such as [MPA] over TCP. The Verbs provide the Consumer with a semantic definition of the RNIC Interface.

RDMA provides Verbs Consumers the capability to control data placement, eliminate data copy operations, and significantly reduce communications overhead and latencies by allowing one Verbs Consumer to directly place information in another Verbs Consumer's memory, while preserving OS and memory protection semantics. Specification of syntactic definitions (API's, hardware registers) and implementation details (hardware, firmware, software tradeoffs) are beyond the scope of this specification.

Section 5 of this document defines the semantics of the RNIC Interface (RI). This interface is implemented as a combination of the RNIC, its associated firmware, and host software. Section 6 describes Queue Pairs, which represent the focus of interaction with the RNIC for work submission. Section 7 describes Memory Management and how the RNIC accesses buffers which contain data to be transferred. Section 8 describes Work Requests and the WR Processing Model, detailing the processing of the units of work from submission to completion. Section 9 describes the RNIC Verbs. The Verbs are an abstract description of the functionality of an RNIC Interface. Section 10 describes security issues associated with implementing an RDMA infrastructure.

A concept frequently encountered in this specification is that of the Verbs Consumer, or simply, the Consumer. The precise meaning of the phrase varies, as a function of context, but it always means the executing entity employing the capabilities of the RNIC to accomplish some objective. In some instances the Verb Consumer may be an OS kernel thread, in others a non-privileged application, and in still others, some special, privileged process. Where the difference is important to the correct behavior of an implementation, it is defined explicitly.

Specification of the API used by the Verbs Consumer to access the capabilities of the RI is outside of the scope of this specification.

Figure 1 is a conceptual diagram that describes an architectural model which includes Privileged Mode consumers, Non-Privileged Mode consumers, RNIC components and the RI.

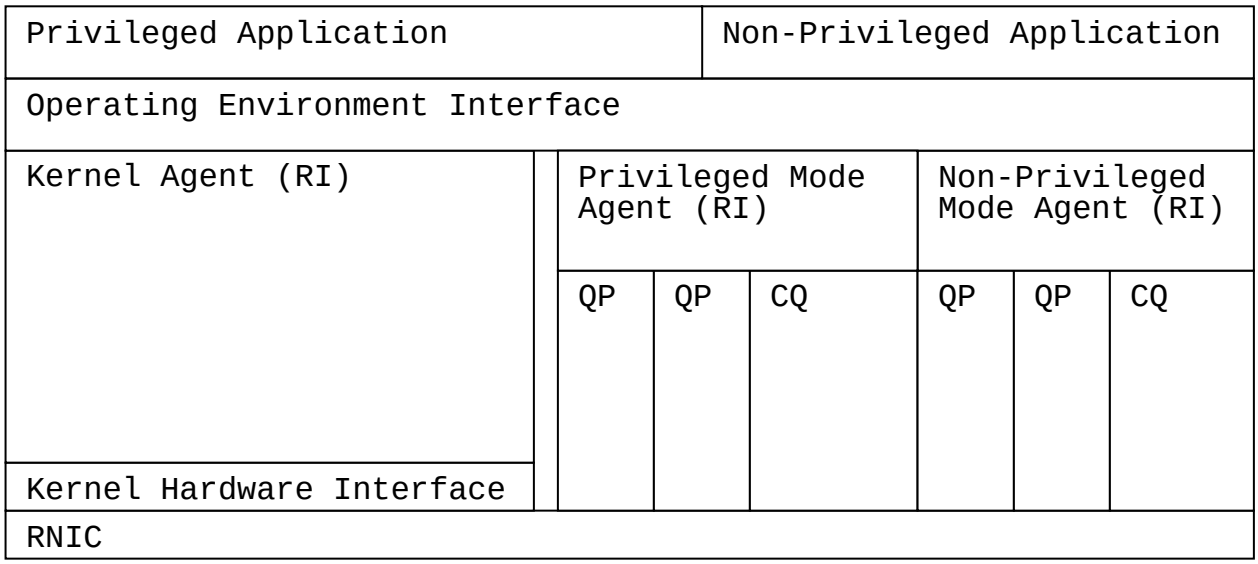


Figure 1 - Architectural RNIC & RI Model

4 Glossary

Access Rights - The Local and Remote Memory Access Rights assigned to an STag. This includes Local Read, Local Write, Remote Read, Remote Write, Remote Access Flag, and Bind.

Address List - A list of addresses that represent the physical pages or blocks referenced by the Physical Buffer List.

Advertisement (Advertised, Advertise, Advertisements, Advertises) - The act of informing a Remote Peer that a Local Node's Buffer is available to it. A Node makes a buffer available for incoming RDMA Read Request Message or incoming RDMA Write Message access by informing its RDMA/DDP peer of the Tagged Buffer identifiers (STag, TO, and buffer length). This advertisement of Tagged Buffer information is not defined by RDMA/DDP and is left to the ULP. A typical method would be for the Local Peer to embed the Tagged Buffer's Steering Tag, TO, and length in a Send Message destined for the Remote Peer.

Affiliated Asynchronous Event - This is an indication from the Verb layer to the Consumer that an event has occurred related to a specific identifiable RNIC Resource, such as a Completion Queue or Queue Pair.

Affiliated Error - An error that can be directly related back to a specific RNIC Resource, such as a QP, S-RQ or CQ, but that cannot be returned through a Work Completion.

Associated QP - The QP on the Remote Peer which is directly accessing the other end of the RDMA Stream.

Asynchronous Error - This is an error that could not be reported through immediate or completion error-handling mechanisms at the local end. An asynchronous mechanism is necessary as a single point of error handling for errors which could not otherwise be reported through the normal mechanism since they are not associated directly with any single QP, S-RQ or CQ or the QP and/or CQ is in a state where an error cannot be reported. Asynchronous errors may be Unaffiliated or may be Affiliated with a specific QP, CQ or S-RQ.

Base Tagged Offset (Base TO) - The offset assigned to the first byte of a Memory Region or a Memory Window.

Bind, Binding, Bound - The act of associating an STag, TO, and Length within a previously registered Memory Region in order to define a Memory Window.

Block List - A list of physical addresses describing a set of memory blocks, which specifies the block size, list of physical addresses, and offset to the start of the memory region of the first block. Each block has the same length and that length can be any value in the range supported by the RNIC. Each block may start at a byte granularity address. The starting address for the entire list may be an offset into the first block and the entire list may have any length.

Complete (Completed, Completion, Completes) - When the Consumer can determine that a particular RDMA Operation has performed all functions specified for the RDMA Operation, including Placement and Delivery. This can be determined through a Work Completion for Signaled Work Requests. For Unsignaled Work Requests, this means that the Completion Rules have been met. Note that this is a superset of the [RDMAP] definition for RDMA Completion.

Completion Error - A Processing Error reported through the Completion Queue.

Completion Queue (CQ) - A sharable queue containing one or more entries which can contain Completion Queue Entries. A CQ is used to create a single point of completion notification for multiple Work Queues. The Work Queues associated with a Completion Queue may be from different QPs and of differing queue types (SQs or RQs).

Completion Queue Entry (CQE) - The RNIC Interface internal representation of a Work Completion.

Completion Status - The resultant status of a Work Request returned as part of a Work Completion.

Consumer, Verbs Consumer - A software process that communicates using RDMA/DDP Verbs. The Consumer typically consists of an application program, or an operating system adaptation layer, which provides some OS specific API.

Direct Data Placement Protocol (DDP) - A wire protocol that supports Direct Data Placement by associating explicit memory buffer placement information with the LLP payload units.

Data Delivery (Delivery, Delivered, Delivers) - Delivery is defined as the process of informing the ULP or Consumer that a particular Message is available for use. This is specifically different from Data Placement, which may generally occur in any order, while the order of Data Delivery is strictly defined.

Data Placement (Placement, Placed, Places) - A mechanism whereby ULP data contained within RDMA/DDP Segments may be put directly into

its final destination in memory without processing by the ULP. This may occur even when the RDMA/DDP Segments arrive out of order. Note that this differs from Data Delivery (see definition in this section). From the Verbs viewpoint, Data Placement is only confirmed upon Completion.

Data Sink - The peer receiving a data payload. Note that the Data Sink can be required to both send and receive RDMA/DDP Messages to transfer a data payload.

Data Source - The peer sending a data payload. Note that the Data Source can be required to both send and receive RDMA/DDP Messages to transfer a data payload.

Event - An indication provided by the RDMAP Layer to the ULP to indicate a Completion or other condition requiring immediate attention.

Fabric - The collection of links, switches, and routers that connect a set of Nodes with RDMA/DDP protocol implementations.

First Byte Offset (FBO) - The offset into the first Physical Buffer of a Memory Region. The value of the FBO cannot exceed the size of the Physical Buffer Entry Size associated with the Memory Region.

Handle - An opaque identifier used to reference an RNIC or an RNIC Resource. Whether this is an index, object or some other construct is outside the scope of this specification.

Immediate Error – An error discovered by the RNIC Interface (RI) and reported through the RI without affecting the RNIC.

Inbound RDMA Read Queue Depth (IRD) - The maximum number of incoming outstanding RDMA Read Request Messages the RNIC’s QP can handle at the Data Source.

Inbound RDMA Read Request Queue (IRRQ) - The RI internal resource which handles incoming RDMA Read Request Messages, queues them for processing them by the RI, and then generates the RDMA Read Response Messages. This corresponds to Queue Number 1 in [DDP].

Invalidate STag (Invalidate, Invalidated, etc.) - A mechanism used to prevent the Remote Peer from reusing an Advertised STag, until the Local Peer transitions the STag to the Valid state.

Invalidate Local STag - A Work Request that takes an STag which is valid within the local RI and performs an Invalidate STag operation.

iWARP - A suite of wire protocols comprised of [RDMAP] & [DDP]. The iWARP protocol suite may be layered above [MPA] and [TCP], or it may be layered over [SCTP] or other transport protocols.

Local Access - The rights used to verify the RNIC's ability to access the Data Sink for incoming Untagged Messages, the Data Source for outgoing Untagged Messages and the Data Source for outgoing RDMA Write Messages.

Local Fence - To block the current operation from executing until all prior local operations submitted on the same Work Queue have Completed.

Local Peer - The RDMA/DDP protocol implementation on the local end of the connection. Used to refer to the local entity when describing a protocol exchange or other interaction between two Nodes.

Lower Layer Protocol (LLP) - The protocol layer beneath the protocol layer currently being referenced. For example, for DDP the LLP is SCTP, MPA, or other transport protocols. For RDMA, the LLP is DDP.

LLP Closed (LLP Close)- When the LLP Stream can no longer be used for data transmission. If there is a single LLP Stream on an LLP Connection, it may also mean that the LLP Connection has been torn down. For example, for TCP this could include the states TIME_WAIT, CLOSING, LAST-ACK, and CLOSED

LLP Connection - Corresponds to an LLP transport-level connection between the peer LLP layers on two nodes.

LLP Reset - The abnormal LLP closing mechanism, usually used to indicate that the LLP Stream (and possibly Connection) was aborted mid-stream. An example of this would be a TCP connection being closed due to the reception or transmission of a TCP RST on the connection.

LLP Stream - Corresponds to a single bi-directional LLP transport-level association between the peer LLP layers on two Nodes. One or more LLP Streams may map to a single transport-level LLP Connection. For transport protocols that support multiple Streams per connection (e.g. SCTP), a LLP Stream corresponds to one transport-level Stream.

Memory Region (MR) - An area of memory that the Consumer wants the RNIC to be able to (locally or locally and remotely) access directly in a logically contiguous fashion. A Memory Region is identified by an STag, a Base TO, and a length. A Memory Region is associated with a Physical Buffer List through the STag.

Memory Registration (Registration, Register) - The mechanism used to enable direct (local or local and remote) access by the RNIC of a Consumer Memory Region. The memory registration operation associates a Physical Buffer List to the Steering Tag (STag) returned.

Memory Translation and Protection Table(s) (TPT) - The data structure(s) used by an RNIC to control buffer access and translate STags and Tagged Offsets into local memory addresses directly accessible by the local Node.

Memory Window (MW) – A subset of a Memory Region, which can be remotely accessed in a logically contiguous fashion. A Memory Window is identified by an STag, a Base TO, and a length, but also references an underlying Memory Region and has Access Rights.

Message Sequence Number (MSN) - For the Untagged Buffer Model, it specifies a sequence number that is increasing with each DDP Message.

Modifiers - In a Verb definition, the list of input and output objects that specify how, and on what, the Verb is to be executed.

Node - A computing device attached to one or more links of a Fabric (network). A Node in this context does not refer to a specific application or protocol instantiation running on the computer. A Node may consist of one or more RNICs installed in a host computer.

Non-Privileged Mode - An operating mode in which Consumers must rely on another agent, having a sufficiently high level of privilege, to manipulate OS data structures.

Non-Shared Memory Region - A Memory Region that solely owns the Physical Buffer List associated with the Memory Region. Specifically, the PBL is not shared, and has never been shared, with another Memory Region.

Outbound RDMA Read Queue Depth (ORD) - The maximum number of outstanding RDMA Read Request Messages the RNIC can initiate from the SQ at the Data Sink.

Outstanding - The state of a Work Request after it has been posted on a Work Queue, but before the retrieval of the Work Completion, or confirmation that the WR has been completed, by the Consumer.

Page List - A list of physical addresses describing a set of memory pages, which specifies the page size, list of physical addresses, and offset to the start of the memory region of the first page. The starting physical addresses of each page is aligned on power-of-two addresses and the size of the page is a power of two. Note that it is possible for the starting offset to be an offset into the first page and to be of a byte granularity and the entire list may have an arbitrary length.

Physical Address - A physical address is used by an RNIC to retrieve contents from the local host's memory. Physical addresses are determined via the translation of the STag and Tagged Offset by the use of the Memory Translation and Protection Table(s).

Physical Buffer - A set of physically contiguous memory locations that can be directly accessed by the RNIC through Physical Addresses. A Physical Buffer can either be a block buffer or a page buffer, depending on its use as part of a Page List or Buffer List.

Physical Buffer Entry Size - The size, in bytes, of each Physical Buffer in the Physical Buffer List. If the Physical Buffer List references a Page List, the size is a power of two. If the Physical Buffer List references a Block List, the size can have any value within the range supported by the RNIC.

Physical Buffer List (PBL) - A list of Physical Buffers. The Physical Buffer List can either be a Block List or a Page List.

Physical Memory Addresses - The addresses an RNIC uses when accessing host system memory.

Pinning memory - A function supplied by the OS that forces the Memory Region to be resident in physical memory and keeps the virtual-to-physical address translations constant from the RNIC's point of view.

Place - Also Placed, Placement. See Data Placement.

Post Receive Queue Work Request (PostRQ) - A Verb that posts a Work Request to the Receive Queue of a Queue Pair. This is done to indicate the Data Sink Buffers for incoming Send Operation Types.

Post Send Queue Work Request (PostSQ) - A Verb that posts a Work Request to the Send Queue of a Queue Pair. This is done to initiate all data transfer operations as well as Fast-Register, Bind MW and Local Invalidate operations.

Privileged Mode - A mode in which Consumers operate where they have a privilege level sufficient to access OS internal data structured directly, and that have the responsibility to control access to the RI.

Processing Error - An error detected below the RNIC Interface during the processing of a Work Request or an incoming RDMA operation.

Protection Domain (PD) – A mechanism for tracking the association of Queue Pairs, Memory Windows, and Memory Regions. PDs are intended to be set by a Privileged Consumer to provide protection of one process from accessing another's memory through the use of the RNIC.

Protection Domain ID (PD ID) - The identifier which represents a Protection Domain. It is passed in as an Input Modifier when creating QPs, Memory Windows and MRs. The value of PD IDs are compared during processing of Work Requests.

Queue Pair (QP) - The pair of queues that allow the Consumer to interact with the RNIC Interface. The two queues are the Send Queue and the Receive Queue. Each queue stores a Work Queue Element from the time it is posted until the time it is completed.

Queue Pair Context - The collection of information needed by the RNIC Interface to perform the RDMA Operations associated with the Queue Pair. This includes various pointers to buffers, queues, and CQs, as well as LLP specific connection and stream information.

Queue Pair Identifier (QP ID) - An identifier representing a Queue Pair.

Read Fence - To block the current operation from executing until all prior RDMA Read Type WRs submitted to the Send Queue have Completed.

Receive Queue (RQ) - One of the two Work Queues associated with a Queue Pair. The Receive Queue contains Work Queue Elements that describe the Buffers into which data from incoming Send Operation Types is placed.

Remote Access - The Access Rights used to verify the RNIC's ability to access the Data Sink for incoming DDP Tagged Messages and the Data Source for RDMA Read Request Messages.

Remote Direct Memory Access (RDMA) - A method of accessing memory on a remote system in which the local system specifies the remote location of the data to be transferred. Employing an RNIC in the

remote system allows the access to take place without interrupting the processing of the CPU(s) on the system. Also used to indicate the layer implementing the RDMAP wire protocol semantics.

RDMA Message - The sequence of DDP segments which represents an RDMA Operation.

RDMA Operation - A sequence of RDMAP Messages, including control Messages, to transfer data from a Data Source to a Data Sink. The following RDMA Operations are defined - RDMA Write Operation, RDMA Read Operation, Send Operation, Send with Invalidate Operation, Send with Solicited Event Operation, Send with Solicited Event & Invalidate Operation, and Terminate Operation. Note that the various forms of Send Operations are defined in [RDMAP] to be called Send Type Operations.

RDMA Protocol (RDMAP) - A wire protocol that supports RDMA Operations to transfer ULP data between a Local Peer and the Remote Peer. See [RDMAP].

RDMA Read Operation - An RDMA Operation that consists of a single RDMA Read Request Message and a single RDMA Read Response Message. The Data Sink uses this operation to transfer the contents of a Data Source buffer from the Remote Peer to the Local Peer.

RDMA Read Request - An RDMA Message used by the Data Sink to request the Data Source to transfer the contents of a buffer. The RDMA Read Request Message describes both the Data Source and Data Sink buffers.

RDMA Read Response - An RDMA Message used by the Data Source to respond to an RDMA Read Request Message.

RDMA Read Type Work Request – A PostSQ Work Request which specifies an operation type of either an RDMA Read or an RDMA Read with Invalidate Local STag.

RDMA Stream - A single bi-directional association between the peer RDMA layers on two Nodes over a single LLP Stream.

RDMA Write Operation - An RDMA Operation that transfers the contents of a source buffer from the Local Peer to a destination buffer at the Remote Peer using an RDMAP Write Message. The RDMAP Write Message only describes the Data Sink's buffer.

RDMA Network Interface Controller (RNIC) - A network I/O adapter or embedded controller with iWARP and Verbs functionality.

Remote Peer - The RDMA protocol implementation on the opposite end of the connection. Used to refer to the remote entity when describing protocol exchanges or other interactions between two Nodes.

Remote RDMA Read Operation – a sequence of events that begins upon receipt of an incoming RDMA Read Request by the RI and stays in-process until the corresponding RDMA Read Response Message has been generated. This includes posting the RDMA Read Request to the Inbound RDMA Read Request Queue (See Section 6.5 - Outstanding RDMA Read Resource Management).

RNIC Interface (RI) - The presentation of the RNIC to the Verbs Consumer as implemented through the combination of the RNIC and the RNIC device driver.

Scatter/Gather Element (SGE) - An individual entry in a Scatter/Gather List. Each SGE consists of an STag, Tagged Offset and Length.

Scatter/Gather List (SGL) - A List of Scatter/Gather Elements. The list describes one or more ULP Buffers which will have their data gathered on transmission or scattered upon reception.

Send - An RDMA Operation that transfers the contents of an Untagged buffer from the Local Peer to an Untagged buffer at the Remote Peer.

Send Operation Types - The set of Send operations that result in the consumption of a Receive Queue Work Request at the Data Sink. Specifically this includes Send, Send with Invalidate, Send with Solicited Event and Send with Solicited Event & Invalidate.

Send Queue (SQ) - One of the two Work Queues associated with a Queue Pair. The Send Queue contains PostSQ Work Queue Elements that have specific operation types, such as Send Type, RDMA Write, or RDMA Read Type Operations, as well as STag operations such as Bind and Invalidate.

Shared Memory Region - An MR that currently shares, or at one time shared, the Physical Buffer List associated with the Memory Region. Specifically, the PBL is currently shared or was previously shared with another Memory Region.

Shared Receive Queue - An optional mechanism which allows the Receive Queues from multiple QPs to retrieve Receive Queue Work Queue Elements from the same shared queue as needed.

Signaled - A WR which requires that the RNIC generate a Work Completion.

Solicited Event (SE) - A facility by which an RDMA Operation sender may cause an Event to be generated at the recipient, if the recipient is configured to generate such an Event, when a Send with Solicited Event or Send with Solicited Event & Invalidate Message is received.

Steering Tag (STag) - An identifier of a Memory Window or Memory Region. STags are composed of two components: an STag Index and an STag Key. The Consumer forms the STag by combining the STag Index with the STag Key. This specification further refines the definitions of STags contained in [RDMAP] and [DDP].

STag Key - The least significant 8 bit portion of an STag. This field of an STag can be set to any value by the Consumer when performing a Memory Registration operation, such as Bind Memory Window, Fast-Register Memory Region and Register Memory Region.

STag Index - The most significant 24 bits of an STag. This field of the STag is managed by the RI and is treated as an opaque object by the Consumer.

Tagged Buffer - A buffer that can be Advertised to a Remote Peer through exchange of an STag, Tagged Offset, and length.

Tagged Offset (TO) - The offset within a Tagged Buffer.

Terminate - An RDMA Message used by a Node to pass an error indication to the Remote Peer on an RDMA Stream.

Upper Layer Protocol (ULP) - The protocol layer above the Verb layer. An example is SDP.

ULP Buffer - A buffer owned above the RI that can be represented within the RNIC, in whole or in part, by a Memory Window or a Memory Region.

ULP Message - The ULP data that is handed to a specific protocol layer for transmission. Data boundaries are preserved as they are transmitted through iWARP.

ULP Payload - The portion of a ULP Message that is contained within a single protocol segment or packet (e.g. a DDP Segment).

Unaffiliated Asynchronous Event - This is an indication from the Verb layer to the Consumer that an event has occurred unrelated to any single identifiable RNIC Resource.

Unsignaled - A Work Request which only generates a Work Completion if it encounters an error during processing.

Untagged Buffer - A buffer which is not Advertised to a Remote Peer, that has Local Access Rights, and that is referenced by an STag, Tagged Offset, and length.

Verbs - An abstract description of the functionality of an RNIC Interface. The OS may expose some or all of this functionality via one or more APIs to applications. The OS will also use some of the functionality to manage the RNIC Interface.

Virtual Address - An address represented in the address space of a local process on a node. It is generally used to present logically contiguous addressability for an underlying and possibly non-contiguous list of physical pages.

Virtual Address Based Tagged Offset (VA Based TO) - The Base TO of an MR or MW that starts at a non-zero TO.

Work Completion (WC) - The output modifiers that the Consumer retrieves from a Completion Queue indicating the results of a Work Request.

Work Queue (WQ) - One of either a Send Queue or Receive Queue.

Work Queue Element (WQE) - The RNIC Interface's internal representation of Work Request.

Work Request (WR) - An elementary object used by Consumers to enqueue a requested operation (WQEs) onto the Send and Receive Queues of a QP.

Work Request List (WRL) - A list of Work Requests.

Zero Based Tagged Offset (Zero Based TO) - The Base TO of an MR or MW that starts at TO=0.

4.1 Abbreviations

CQ - Completion Queue

CQE - Completion Queue Entry

DDP - Direct Data Placement Protocol

FBO - First Byte Offset

IRD - Inbound RDMA Read Queue Depth

IRRQ - Inbound RDMA Read Request Queue

LLP - Lower Layer Protocol

MR - Memory Region

MW - Memory Window

ORD - Outbound RDMA Read Queue Depth

PBL - Physical Buffer List

PD - Protection Domain

PD ID - Protection Domain Identifier

QP - Queue Pair

QP ID - Queue Pair Identifier

RQ - Receive Queue

RDMA - Remote Direct Memory Access

RDMAP - Remote Direct Memory Access Protocol

RNIC - RDMA NIC

RI - RNIC Interface

SGE - Scatter-Gather Element

SGL - Scatter-Gather List

SE - Solicited Event

S-RQ - Shared Receive Queue

SQ - Send Queue

STag - Steering Tag

TO - Tagged Offset

TPT - Translation & Protection Table

ULP - Upper Layer Protocol

WC - Work Completion

WQ - Work Queue

WQE - Work Queue Element

WR - Work Request

WRL - Work Request List

5 RNIC Interface

The RNIC Interface (RI) is the locus of interaction between the Consumer of RNIC services and the RNIC. Semantic behavior of the RNIC is specified via Verbs, which enable creation and management of Queue Pairs, management of the RNIC, management of Work Requests, and transferring error indications from the RI that may be surfaced via the Verbs. All these activities must be carried out so as to enable Verbs Consumers to expect the same level of protection and security as are guaranteed other entities supported by the host operating system.

A fundamental function of the RI is management of RNICs. This includes arranging access to them, accessing and modifying their attributes, and shutting them down. These activities are described below, and details of the corresponding Verbs semantics are given in subsequent sections.

Direct, protected access to Consumer memory is critical to realizing the performance potential of the RNIC. This specification describes the semantics of memory access defined in this architecture. It describes in detail the ideas of Memory Regions and Memory Windows, how they are created and managed, Access Rights for local and remote access to registered memory, and the semantics of errors that may arise.

The RI is assumed to be a traditional software interface, typically synchronous in behavior, while QP interactions are assumed to be work requests queued to connection specific, hardware based queues. The queue processing model and associated memory protection semantics allow QPs to be safely mapped and utilized by both Non-Privileged and Privileged routines.

Queue Pairs (QPs) are a key component required for the operation of the RI. They are the RNIC resource used by Consumers to submit Work Requests to the RI. A QP is used to interact with an RDMA Stream on an RNIC which is running the RDMA Protocol. There may be thousands of QPs per RNIC. Each QP provides the Consumer with a single point of access to an individual RDMA Stream.

Work Requests (WRs) provide the mechanism for Consumers to enqueue Work Queue Elements (WQEs) onto the Send and Receive queues of a QP. The varieties of WRs, and the dynamics of their creation, use, and disposition are described in the sections to follow, as are the disposition of errors that may arise as WR are processed. Details of the WR contents are discussed as well.

Completion Queues (CQs) provide the mechanism for the Consumer to retrieve WR status. In addition, there are notification mechanisms

which help a Consumer to efficiently notice when WRs have completed processing in the RI. There may be thousands of CQs per RNIC.

Event Handlers provide the mechanism for Consumers to be notified of Asynchronous Events which occur within the RI but which cannot be reported through the Completion Queues due to their asynchronous nature or the fact that they are not easily associated with a Work Completion.

5.1 The RNIC

Consumers gain access to an RNIC through the RNIC Interface. The Verbs allow the Consumer to open the RNIC, retrieve RNIC attributes, and close the RNIC.

All resources MUST be in the scope of the RNIC on which they are created. This means that there is no requirement for resources on one RNIC to be available, associated with or meaningful to another RNIC, even if they are managed by the same RNIC driver. This includes all QPs, STags, PDs, CQs, and multiple Completion Event Handlers. This also means that any IDs which are created by the RI are specific to that RNIC and are not guaranteed to be unique across all RNICs.

An intent of the architecture is to allow an implementation to pass Work Requests and Work Completions to and from a Non-Privileged Mode Consumer process directly to and from the RNIC. Another intent of the architecture is to optimize for a Privileged Mode implementation, which shares the Work Request and Work Completion requirements of Non-Privileged Mode Consumers but has slightly different memory management requirements.

Because the architecture attempts to optimize for both Privileged Mode and Non-Privileged Mode Consumers, there are some Verbs and Verb modes which are not allowed to be executed by non-Privileged Mode Consumers. An example of this is the use of the STag of zero or the ability to do Fast-Register WRs. In addition, there are some operations that, while being allowed in kernel mode, are intended to be used by Non-Privileged mode applications. An example of this is Memory Windows. Any restrictions are clearly specified in this document where required.

5.1.1 RNIC Resources

RNIC Resources can be allocated from a variety of places. They can be allocated in host memory on behalf of the Consumer or allocated within the RNIC. Where an RNIC allocates resources is implementation specific. Consequently, values that the RNIC returns as output modifiers when Querying the RNIC indicate the maximum amount of any

given resource it can allocate, in the absence of other resource allocations.

For example, an RNIC may allocate QPs, CQs from the same memory within the RNIC. If a Consumer allocates the maximum amount of QPs before allocating any CQs, it may not be able to allocate any CQs due to an insufficient resource condition - even though the RNIC indicates that its maximum number of CQs is much larger than the number currently allocated.

The purpose of a handle is to provide a mechanism to lookup a specific resource. Resources that have handles associated with them are the RNIC, CQ, S-RQ, QP, and Asynchronous Event Handler. Often a handle is an address in memory. An identifier or index also references a specific resource. An identifier or index is used when the value must be used in a comparison operation. The QP ID, PD ID, Completion Event Handler Identifier and STag Index fall in this category.

It is expected that a resource manager above the RI will manage RNIC resources appropriately for the operating environment.

5.1.1.1 Expected Creation Sequence

Due to RI Resource interdependencies, there is an ordering sequence to the allocation and creation of RNIC resources. The sequence indicated below, while not strictly required in all cases, may be helpful to the reader.

1. Open the RNIC and setup up an Asynchronous Event Handler.
2. Prior to initiating a LLP Connection, select the opened RNIC on which you will create the connection and create a Protection Domain.
3. Create one or more Completion Queues.
4. Set up one or more Completion Event Handlers.
5. Allocate and initialize a Shared Receive Queue, if desired.
6. Allocate and initialize one or more QPs.
7. Register one or more Memory Regions.
8. Allocate Non-Shared Memory Region STags, if desired.
9. Allocate Memory Windows, if desired.
10. Transition the QP through the state diagram to RTS.

11. Initiate Work Request Processing through PostSQ, PostRQ and Poll CQ.

Below in Figure 2 is a dependency diagram which may also be helpful when determining the order in which resources are created. The arrows indicate that the resource the arrow comes from must be created or allocated before the item the arrow points to can be created or allocated.

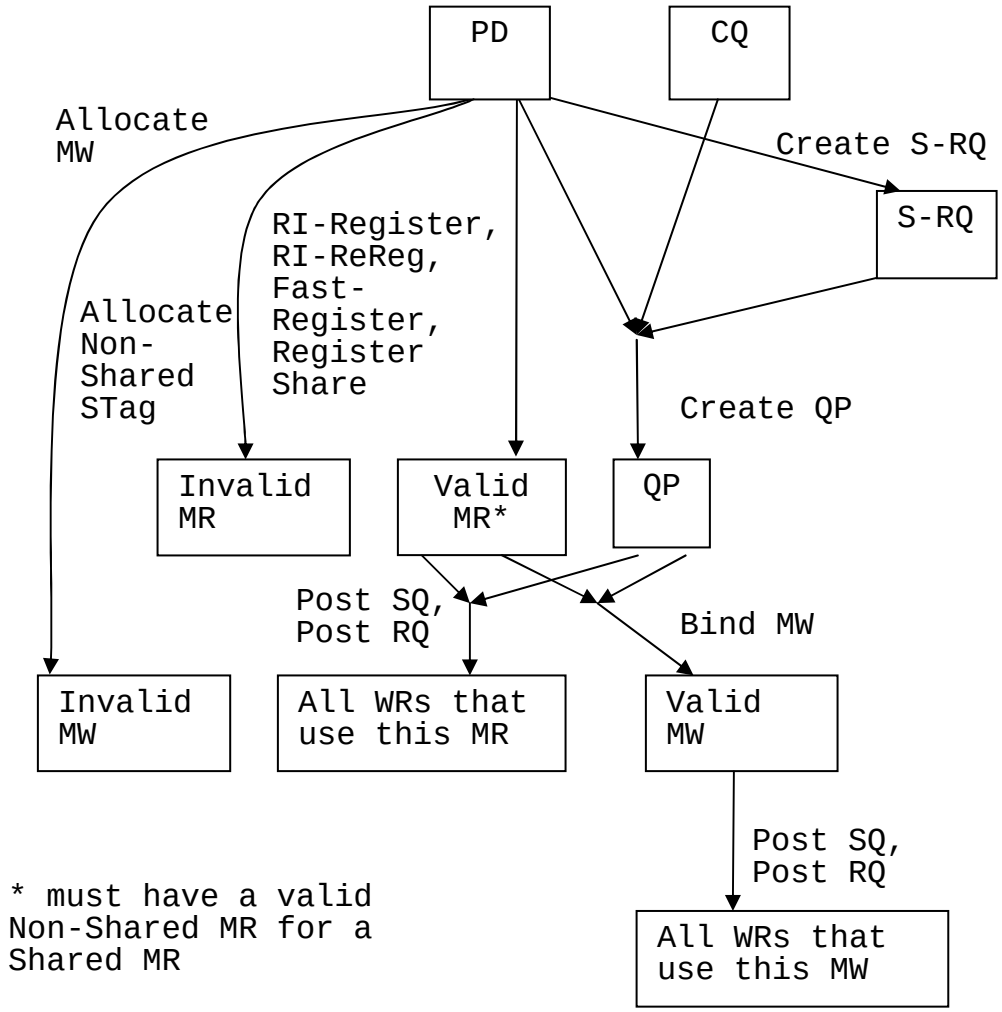


Figure 2 - Resource Creation Dependency Diagram

5.1.1.2 Expected Destruction Sequence

Due to RI Resource interdependencies, there is an ordering of de-allocation and destruction of RNIC resources. The sequence indicated

below, while not strictly required in all cases, may be helpful to the reader.

1. Invalidate all Memory Windows which are in the Valid state through a QP WR, if possible.

2. Drain the SQ & RQ of WRs and poll the Work Completions through the CQ.

3. Transition the QP state to Closing.

4. When the QP is in the Idle state, Destroy the Memory Windows.

5. Destroy the Memory Regions.

6. Destroy the Queue Pair.

7. Destroy the Shared Receive Queue, if created.

8. Destroy the Completion Queues.

9. Destroy the Protection Domain.

10. Close the RNIC.

Below in Figure 3 is a dependency diagram which may also be helpful when determining the order in which resources are destroyed. The arrows indicate that the resource the arrow comes from must be destroyed or deallocated before the item the arrow points to can be destroyed or deallocated. A dashed line means the action should occur before the resource can be destroyed. A solid line means the action must occur before the resource can be destroyed.

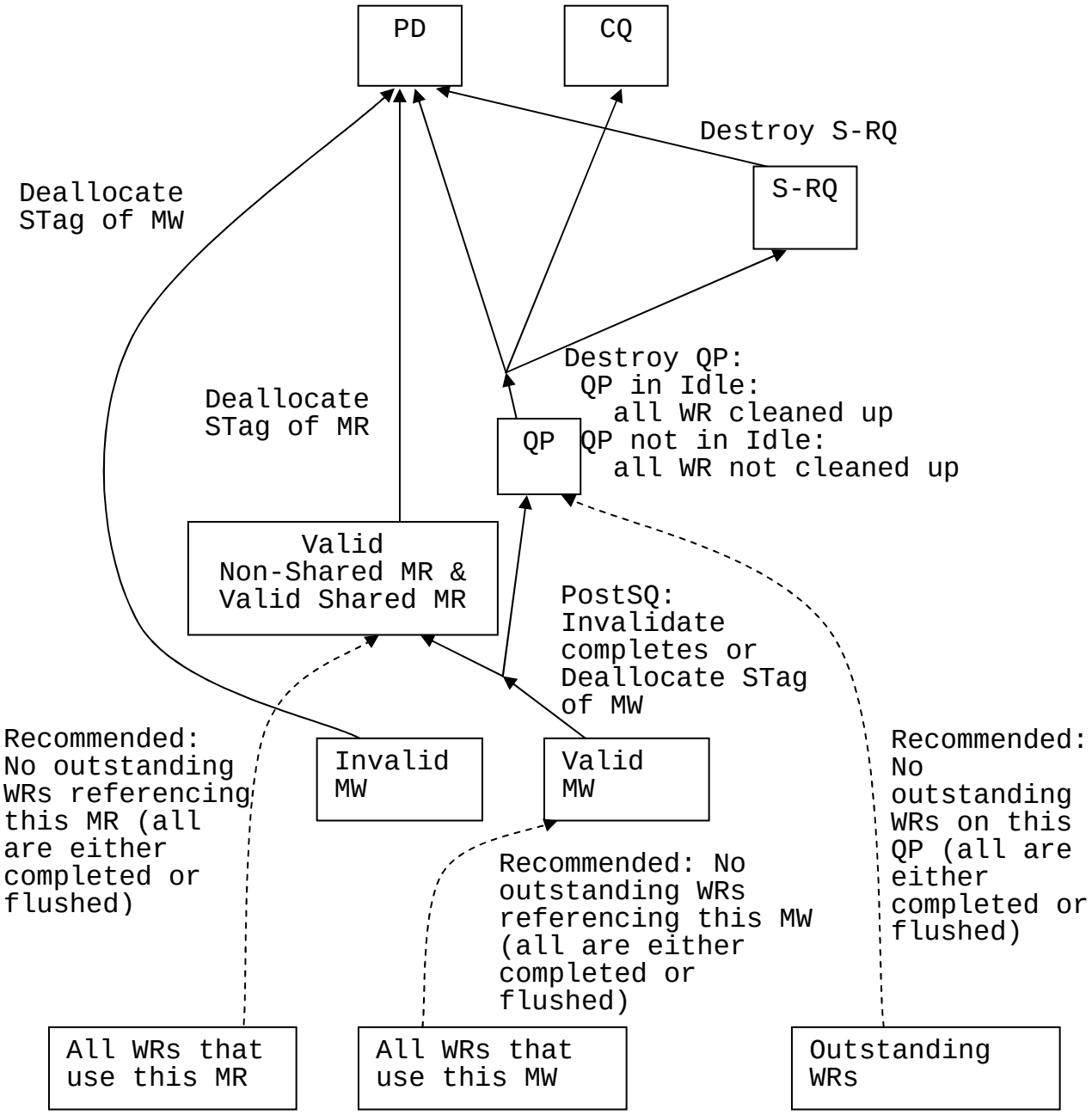


Figure 3 - Resource Destruction Dependency Diagram

5.1.2 Opening an RNIC

The Open RNIC Verb is used to open an RNIC and returns an opaque handle to uniquely reference each RNIC so that Consumers can distinguish between RNICs in the Local Node.

Opening an RNIC prepares it for use by the Consumer. Once opened, an RNIC cannot be opened again until after it has been closed. At the time the RNIC is opened, the RI MUST perform any initialization functions required by the RNIC and the RI.

When the Consumer invokes the Open RNIC Verb, it indicates if this RNIC is to be opened in Page Mode or Block Mode. The RI MUST initialize the RNIC in either Page Mode or Block Mode, as indicated by the Consumer with the input modifier. This will affect all Memory Registrations and usage as well as resource consumption on the RNIC. Note that while Page Mode MUST be supported, Block Mode is OPTIONAL. For more information on Block Mode vs. Page Mode, see Section 7.6.2 - Physical Buffer Lists.

Detailed information on the accompanying Verb can be found in Section 9.2.1.1 - Open RNIC.

5.1.3 Query RNIC

Consumers MUST be able to retrieve all of the defined attributes and characteristics of the RNIC through the Query RNIC Verb. The full list of RNIC Attributes is defined in Section 9.2.1.2 - Query RNIC.

The maximum values returned when querying the RNIC are values which the RI will not exceed. This does not imply that a Consumer can allocate all resources to their maximum levels simultaneously.

5.1.4 Closing an RNIC

Closing the RNIC resets the RNIC and deallocates any resources allocated during the RNIC open.

The RI MUST track all RNIC resources created on behalf of the Consumer, such as those allocated within the RI during the creation of PDs, QPs, CQs, Memory Windows and MRs. When the Close RNIC verb returns, the RI MUST have freed all RNIC resources.

Detailed information on the accompanying Verb can be found in Section 9.2.1.3 - Close RNIC.

5.2 Protection Domains

A Protection Domain (PD) is the mechanism used to associate Queue Pairs with Memory Regions and Memory Windows as a means of enabling

and controlling RNIC access to host system memory. A Protection Domain is represented by a unique identifier called a Protection Domain Identifier (PD ID).

When the Consumer creates a PD, a PD ID is returned. The Consumer then provides the PD ID to the RI when creating QPs, MRs & Memory Windows. When a data transfer takes place, if the STag refers to an MR, then the PD ID of the MR is validated against the PD ID of the QP. If they do not match, the data transfer generates an error and no data transfer takes place. If the STag refers to an MW, then the PD ID of the MW is validated against the PD ID of the QP when the MW is Bound to the QP. When a data transfer takes place, the QP ID of the MW is validated against the QP ID of the QP. These rules allow the Consumer to ensure that any STag being used on that connection, either locally or remotely, has been specifically allowed by the Consumer to be used on that connection.

Each Queue Pair in an RNIC MUST be associated with a single PD ID. Multiple Queue Pairs MUST be able to be associated with the same PD ID.

Each Memory Region MUST be associated with a single PD ID. Multiple Memory Regions MUST be able to be associated with the same PD ID.

Each Memory Window MUST be associated with a single PD ID when allocated. Multiple Memory Windows MUST be able to be associated with the same PD ID.

The RI MUST be able to associate any PD ID with any MW, MR, QP or S-RQ on the RNIC.

Binding a Memory Window to a Memory Region and Fast-Register are performed as Send Queue operations. The Bind operation MUST only be allowed if the PD ID of the QP matches the PD ID of the Memory Region and the PD ID of the QP matches the PD ID of the Memory Window. Similarly, the Fast-Register operation MUST only be allowed if the PD ID of the QP matches the PD ID of the STag used as an input modifier for the Fast-Register. If the PD ID checks fail for either operation, the operation MUST NOT take place and a Completion Error MUST be generated.

Note that S-RQs use PDs as well. PD rules for S-RQs are covered in Section 6.3

5.2.1 Allocating a PD

Protection Domains MUST only be allocated through the RI. A PD ID is required to be supplied as an input modifier when creating a Queue Pair, registering a Memory Region, or allocating a Memory Window.

The RI MUST assign a unique PD ID to each PD allocated by the RI. PD ID's MUST be unique per RNIC. PD ID's MAY be unique across multiple RNICS which share the same RI.

Detailed information on the accompanying Verb can be found in Section 9.2.2.1 - Allocate PD.

5.2.2 Deallocating a PD

PDs MUST only be deallocated through the RI. A PD MUST NOT be deallocated if it is still associated with any Queue Pair, Shared Receive Queue, Memory Region, or Memory Window. If this is attempted, the Verbs MUST return an Immediate Error and not allow the PD to be deallocated.

Detailed information on the accompanying Verb can be found in Section 9.2.2.2 - Deallocate PD.

5.3 Completion Queues

The Completion Queue consists of entries to hold Work Completions. The RI's internal representations of Work Completions are called Completion Queue Entries (CQEs). The RI will post a CQE to the CQ when it completes the operation of a Signaled WR. The Consumer then Polls the CQ to retrieve the CQE as a Work Completion. When the Work Completion is retrieved, the CQE is freed from the CQ and the entry is available for another Work Request's Work Completion information. For an Unsignaled WR, the RI will not generate a CQE when the WR completes successfully. The RI will post a CQE to the CQ when an Unsignaled WR completes in an error. For more information on Signaled and Unsignaled Completions, see Section 8.1.3.1.

A Completion Queue (CQ) MUST be the only mechanism used for the retrieval of Work Completions.

A single CQ is used to hold CQEs from one or more Work Queues across one or more Queue Pairs on the same RNIC. A CQ MAY have zero or more Work Queue associations. Completion Queues MUST be able to service Send Queues, Receive Queues or both. Work Queues from multiple QPs MUST be able to be associated with a single CQ.

Completion Queues and Completion Queue Entries are internal to the RNIC Interface, and are not directly accessible, nor is the format directly visible, by Verb Consumers.

5.3.1 Creating a Completion Queue

Completion Queues MUST only be created through the RNIC Interface.

The RI MUST verify that the Consumer has specified the number of CQEs the CQ should hold when creating a Completion Queue. The Consumer should ensure that this value is the maximum number of Completions the Consumer expects to be outstanding. The RNIC will then create the CQ with at least the specified number of entries. The number of entries allocated for the CQ by the RI MAY be greater than the number requested. If the CQ can be created, the RI MUST return the actual number of entries allocated for that CQ to the Consumer. If the RI is unable to allocate at least as many entries as the Consumer requested, an Immediate Error MUST be returned and the CQ MUST NOT be created.

The RI is NOT REQUIRED to perform CQ overflow detection or protection. Therefore, the CQ overflow error codes in this document are OPTIONAL. When an overflow occurs, the results are indeterminate. Overflow of a CQ MUST NOT affect QPs which do not report Work Completions to that CQ and MUST NOT affect other CQs. Consequently, when creating the CQ, the Consumer should request enough outstanding Work Requests so that if every possible outstanding WR were to complete (such as may happen in an error case), there would be room for the CQE on the CQ. The RI MUST NOT enforce that every WQE on every Work Queue associated with the CQ must have a CQE available for the WQE's Work Completion information.

If the Consumer wishes to have deterministic error behavior, at Create/Modify QP, the sum of the maximum number of WQEs associated with a single CQ should be less than or equal to the number of entries in the CQ. A Consumer can size the CQ smaller, in which case the error semantics of a CQ overflow are not deterministic, but possible RNIC behavior includes overwriting previous CQEs in whole or in part and thus may result in a data integrity issue.

An additional consideration for sizing the CQ is QP Destruction. Any outstanding WRs which were on a Work Queue when it is destroyed may occupy entries on the associated CQ. For more information, see Section 6.1.4 - Destroying a Queue Pair.

Detailed information on the accompanying Verb can be found in Section 9.2.3.1 - Create CQ.

5.3.2 Querying Completion Queue Attributes

There are two Completion Queue attributes that can be queried through the RI.

The first of these attributes is the maximum number of entries allowed on the CQ. This attribute MUST be able to be retrieved through the Query CQ Verb.

The other attribute is the Completion Event Handler Identifier, which also MUST be able to be retrieved through the Query CQ Verb.

With one exception, the CQ Verbs do not expose which Work Queues are associated with a CQ. The exception is that the QP ID is reported by Poll CQ.

Detailed information on the accompanying Verb can be found in Section 9.2.3.2 - Query CQ.

5.3.3 Modifying Completion Queue Attributes

An implementation MUST support resizing of a CQ through the RI while WRs are outstanding. Work Completions MUST NOT be lost due to a CQ resize. Resizing the CQ MUST NOT directly generate errors beyond Resize CQ Verb Immediate Errors and must either succeed or fail atomically. It is understood that this may adversely affect performance, and MAY result in connection timeouts. Note that this could ultimately result in the connection being torn down. If the Consumer wishes to avoid any possibility of a connection being torn down during the CQ resize operation, it should quiesce operations to the Work Queues associated with the CQ before resizing the CQ. The RI MUST NOT allow a CQ to be resized to a size that is smaller than the number of CQEs currently on the CQ; if this is attempted, an Immediate Error MUST be returned.

Detailed information on the accompanying Verb can be found in Section 9.2.3.3 - Modify CQ.

5.3.4 Destroying a Completion Queue

CQs MUST only be destroyed through the RI.

A CQ MUST NOT be destroyed if it is still associated with any Work Queue. If this is attempted, the Verbs MUST return an Immediate Error and not allow the CQ to be destroyed.

When the Destroy CQ Verb returns, the RI MUST have returned or released any host resources allocated below the RNIC Interface on behalf of the Consumer that are related to the specified CQ. After the Destroy CQ Verb returns, the RI MUST NOT return any more Work Completions that are associated with the destroyed CQ.

Detailed information on the accompanying Verb can be found in Section 9.2.3.4 - Destroy CQ.

6 Queue Pairs

Queue Pairs (QP) are the RNIC resource used by Consumers to submit operations to the RNIC. A QP consists of a pair of Work Queues (Send and Receive) as well as a posting mechanism for each queue. The Send Queue (SQ) and Receive Queue (RQ) are each Work Queues, in that the Consumer posts Work Requests (WR) to them in order to get the RI to perform operations. In addition, there are resources that make up the QP with which the Consumer does not directly interact. These include the Inbound RDMA Read Request Queue and the Work Queue Elements (WQEs).

Work Queue Elements are the representation of Work Requests inside of the RI, once the Work Requests have been posted to the QP.

An internal Inbound RDMA Read Request Queue (IRRQ) MUST be associated with a Queue Pair when the QP is created or modified to support greater than zero incoming RDMA Read Request Messages. The IRRQ enqueues incoming RDMA Read Request Messages and processes them in order, sending RDMA Read Response Messages as a result. The depth of this queue MUST be specified when the QP is created and is set with the IRD Input Modifier.

A QP is created by the RI at the request of a Consumer. The resources required by the RI to create the Work Queues and get them to transmit and receive resources are allocated at this time. The memory needed may be allocated from system memory, memory associated within the RNIC, or any other resources accessible through the Verbs.

Certain QP attributes may be changed after QP creation. A Modify QP Verb is provided to modify the attributes. The details of this Verb are defined in Section 6.1.3 - Modifying Queue Pair Attributes.

The Consumer should instruct the RI to destroy a QP that is no longer in use. The semantics for destruction of a QP are provided in this Section 6.1.4 - Destroying a Queue Pair.

The Verbs Post Send Queue Work Request (Section 9.3.1.1 PostSQ) and Post Receive Queue Work Request (Section 9.3.1.2 PostRQ) provide a posting mechanism for the Consumer to indicate to the RI that there is work for the RI to perform and that there is a new WR, represented within the RI by a WQE, on the Work Queue. Details of Work Request handling are defined in Section 8 - Work Requests and the WR Processing Model.

6.1 Queue Pair Resource Handling

6.1.1 Creating a Queue Pair

Queue Pairs are created through the RI. When a QP is created, the RI MUST verify that the Consumer has specified a complete set of initial attributes. The attributes that need to be defined when the QP is created are specified in Section 9.2.5.1 - Create QP.

Two of the attributes that must be initialized when a QP is created is the maximum number of Outstanding WRs on the SQ and the maximum number of Outstanding WRs on the RQ. This number represents the maximum number of WRs which have been submitted but which have not Completed at any given time. This is really the maximum depth of the SQ or RQ and not the number of WRs on the Work Queue at the moment. The RI MUST support Consumers specifying the maximum number of outstanding WRs on the SQ and on the RQ and allow the maximum number of outstanding WRs on the SQ to be different from that on the RQ. The Consumer requests a maximum number of outstanding WRs on the SQ and on the RQ. The RI MUST return the maximum number of outstanding WRs allocated on the SQ and on the RQ, and each of these numbers MAY be greater than the number requested. For information on determining when WRs are completed, see Section 8.1.3.1 - Signaled Completions. Note that if the QP uses an S-RQ for incoming Untagged Messages, the maximum number of Outstanding WRs on the RQ is not needed.

Each Work Queue in a QP MUST be associated with one and only one CQ when that QP is created.

Since both WQEs and CQEs are implemented below the RI and the implementations are outside the scope of this specification, they may be implemented using a variety of mechanisms, including in the Local Host virtual memory address space. The RI MAY require that the Work Queues be in the same memory space as the corresponding Completion Queues or the creation of the QP will fail. Therefore the Consumer should assume that the CQ & QP share the same address space. If the RI detects that QP and CQ are inaccessible to each other, creation of the QP MAY fail.

Other attributes that MUST be initialized when a QP is created are whether or not this QP will support the Fast-Register Non-Shared Memory Region operation and whether the QP supports an STag of zero. These attributes must only be enabled on QPs used by Privileged Mode Consumers. See Section 7.2.1 - STag of zero for an explanation of the STag of Zero. For an explanation of the Fast-Register Non-Shared Memory Region operation, see Section 7.3.2.5 - Fast-Register Non-Shared Memory Region.

When a QP is created it MUST be associated with a PD. This is done by specifying the PD ID as an Input Modifier to Create QP.

An attribute that MUST be created within the RI when the Consumer invokes the Create QP Verbs is a Queue Pair Identifier (QP ID). The QP ID MUST be used by the RI to uniquely identify this QP within this RNIC to the Consumer. The QP ID is used when trying to determine if a Memory Window is Bound to the QP, as discussed in Section 7.10.2 - Binding Memory Windows to Memory Regions. The QP ID value MUST be returned as part of the Create QP, Query QP and Poll CQ Verbs.

Create QP MUST NOT associate an LLP Connection or LLP Stream with the QP. No data will flow until the QP is Associated with another QP through an LLP Stream and the QP state is changed to RTS. For more details, see Section 6.6.1 - Connection Initialization.

A QP can exist in one of several states. For the details of the QP states, see Section 6.2 - Queue Pair Resource States. The following list summarizes the valid QP states:

* Idle state – No LLP Stream is associated with the QP.

* RTS state – An LLP Stream is associated with the QP and normal data transfer can occur.

* Closing state – An error free LLP Close has begun but has not finished. It was initiated by either the Remote Peer or Local Peer.

* Terminate state – An error occurred. A Terminate Message was either sent or received, and the QP is waiting for either a LLP Close or LLP Reset before automatically transitioning the QP to the Error state.

* Error state – An error occurred. No LLP Stream is associated with the QP. A Terminate Message will be available through QueryQP if the QP transitioned through the Terminate state before entering the Error state. If the transition was from the Closing state to the Error state, a Terminate Message may be available.

When the QP is created, it is initialized to the Idle state.

Detailed information on the accompanying Verb can be found in Section 9.2.5.1 - Create QP.

6.1.2 Querying Queue Pair Attributes

Queue Pairs have attributes that can be retrieved through the Query QP Verb. The RI MUST support the complete list of QP attributes as described in Section 9.2.5.2 - Query QP.

6.1.3 Modifying Queue Pair Attributes

Certain QP attributes may be modified after the QP has been created. If the Consumer invokes Modify QP without specifying all Required Attributes as defined in Figure 4, the RI MUST NOT modify any of the QP attributes and MUST return an Immediate Error. The RI MUST allow the Consumer to request a change for the Allowed Additional Attributes as described in Figure 4, for the QP state transitions also shown in the figure. On Consumer request, the RI MAY change the allowed Additional Attributes as described in Figure 5, for the QP state transitions shown in the figure, if the RI indicates through Query RNIC that the attribute in question is allowed to be changed. The Modify QP Verb output modifiers can be used to determine if the changes are actually made.

If any of the QP attributes requested to be modified are invalid or the requested state transition is invalid, the RI MUST NOT modify any of the QP attributes and an Immediate Error MUST be returned. Note that the table is heavily dependent upon the QP state. For further information on the QP state, see Section 6.2 - Queue Pair Resource States.

Transition	Attributes that Consumer is Required to Supply for the State Transition	Attributes that the RI must Support and the Consumer may Supply for the State Transition
Idle->Idle	Next state	ORD
Idle->RTS	Next state, LLP Stream Handle	Stream message buffer, ORD
Idle->Error	Next state	None
RTS->RTS (Footnote 1)	Next state	ORD
RTS->CLOSING	Next state	None
RTS->TERM	Next state	None
RTS->Error	Next state	None
Error->Idle	Next state	None

Figure 4 - Allowable QP Attribute Modifications

Footnote 1: Changing these parameters in RTS requires care to avoid race conditions to prevent errors.

Transition	Attributes that the RI Optionally Supports and the Consumer may Supply for the State Transition
Idle->Idle	Max Number of SQ WQE, Max Number of RQ WQE (Footnote 2), IRD, QP's RQ Limit, QP's RQ Limit Armed
Idle->RTS	Max Number of SQ WQE, Max Number of RQ WQE (Footnote 2), IRD, QP's RQ Limit, QP's RQ Limit Armed
Idle->Error	None
RTS->RTS (Footnote 3)	Max Number of SQ WQE, Max Number of RQ WQE (Footnote 2), IRD
RTS->CLOSING	None
RTS->TERM	None
RTS->Error	None
Error->Idle	None

Figure 5 - Optional QP Attribute Modifications

It is possible to modify the QP attributes in Figure 4 and Figure 5 with Work Requests outstanding on the QP. Depending on the modification, any Work Requests outstanding on the specified QP might not execute properly when the attributes are changed.

An RNIC MAY allow the Consumer to change the maximum number of outstanding WRs on the SQ and on the RQ. The RNIC MUST indicate to the Consumer if it supports the ability to change the number of

---

Footnote 2: Note that changing the Max Number of RQ WQEs has no effect if the QP uses an S-RQ

Footnote 3: Changing these parameters in RTS requires care to avoid race conditions to prevent errors.

outstanding WRs on a QP. If the RNIC supports it, it MUST allow the number of outstanding WRs on both the SQ and the RQ to be changed while WRs are still outstanding. In addition, the RI MUST support the ability to change this on every QP if it indicates an ability to change the outstanding number of WRs.

It is understood that changing the number of WRs that a Work Queue may have outstanding may adversely affect performance. Resizing the QP MUST NOT cause Immediate, Completion or Asynchronous Errors, with the exception of Immediate Errors returned by the Modify Queue Pair Verb and possible LLP time-outs. It is expected that the resize operation MAY adversely affect the Associated QP attempting to communicate with the Local QP during the resize operation in the form of LLP time-outs and retries which could result in LLP Stream teardown (which would result in an Asynchronous Error). It is suggested that the Consumer only perform this resize operation when activity on the connections has been quiesced to minimize the risk of transitioning Associated QPs to the Error state as a result of LLP time-outs.

If the number of requested outstanding WRs is smaller than the actual number of outstanding WRs currently on the Work Queue(s), then the modification of the QP MUST fail with an Immediate Error and the QP MUST remain in the original state.

For information on performing a Modify QP and modifying the value of IRD and/or ORD, see Section 6.5 - Outstanding RDMA Read Resource Management.

When the Modify QP Verb completes, any state change requested MUST have occurred or an Immediate Error MUST be returned, in which case the QP state and accompanying modifier changes MUST remain as they were prior to the Modify QP Verb being invoked.

The LLP Stream and the LLP Stream Message Buffer Input Modifiers for Modify QP are covered in Section 6.6.1.

Detailed information on the accompanying Verb can be found in Section 9.2.5.3 - Modify QP.

6.1.4 Destroying a Queue Pair

Queue Pairs MUST only be destroyed through the RNIC Interface.

Successful destruction of a QP MUST release all resources allocated by the RI for the QP on behalf of the Consumer. The RI MUST have destroyed the QP when the Destroy QP Verb has successfully completed. If the LLP Stream is still associated with the QP, a Destroy QP MUST include disassociating the LLP resources from the QP, and MAY include an LLP Reset. After a Destroy QP finishes, the

QP ID will be immediately available for use on any subsequently created QP. The QP will cease processing all WRs, and no additional CQEs resulting from any outstanding WRs on this QP will be posted to the CQ.

The RI MUST not allow a QP to be destroyed if there are still Memory Windows Bound to the QP. If the Consumer attempts to destroy a QP with Memory Windows Bound to the QP, an Immediate Error MUST be returned by the RI.

The RI MUST allow the Destroy QP Verb to succeed regardless of the QP's state, provided there are no MWs Bound to it. For more information on the resource destruction and deallocation sequence, see Section 5.1.1.2 - Expected Destruction Sequence.

It is RECOMMENDED that before a Consumer attempts to destroy a Queue Pair, it should cleanly complete all outstanding Work Requests and invalidate all Memory Windows which are Bound to the QP. It is recommended that ULPs and Consumers provide a graceful termination mechanism, return all Advertised STags to a known state, submit WRs to Invalidate all outstanding Memory Windows and then move through the Closing state. The Consumer should then retrieve all outstanding Work Completions through the CQ(s) associated with the QP's SQ & RQ. Only then should the Consumer destroy the QP.

A QP is allowed to have Work Requests outstanding on both Work Queues when a request to destroy the QP is made.

Any outstanding WRs posted to the QP but not yet processed by the RI MAY result in CQEs that MAY be retrievable by the Consumer. Note that even in the case where CQEs were generated it might not be possible for the Consumer to retrieve them after the QP has been destroyed. Since it is implementation dependent as to whether CQEs are consumed for outstanding WRs on a QP after that QP is destroyed, for the purposes of CQ overflow prevention, the Consumer should consider each outstanding WR to have consumed an entry in the CQ. There are three ways to free the CQE consumed within the CQ. Any method is acceptable and they are not mutually exclusive. The three methods are:

* the Consumer polls the CQ (See Section 9.3.2.1 - Poll for Completion (Poll CQ)) until the CQ is empty, or

* the Consumer retrieves a WC for a WR which was submitted to a Work Queue associated with the same CQ and that WR was submitted after the previous QP was destroyed, or

* the Consumer polls (See Section 9.3.2.1 - Poll for Completion (Poll CQ) a number of Work Completions equal to the total number of entries that the CQ can hold.

Detailed information on the accompanying Verb can be found in Section 9.2.5.4 - Destroy QP.

6.2 Queue Pair Resource States

The RI MUST restrict the QP to only be in one of the five Resource States (or just "states") as shown in Figure 6. The RI MUST NOT support transitions between QP states that are not shown in Figure 6.

During any state in which iWARP processing is done, it is possible for errors to be detected by the RNIC. When this occurs, the QP state will eventually transition to the Error state.

State transitions must only be initiated by the Modify QP Verb, except where otherwise explicitly stated in the state descriptions.

Creation of a QP causes the QP to enter the state diagram in the Idle state. Destruction of a QP causes the QP to exit the state diagram.

Below, in Figure 6, is the QP State diagram. It shows the five QP states and the allowed transitions between states as well as the events and methods which cause those transitions. The individual states and transitions are described in the following sections in detail.

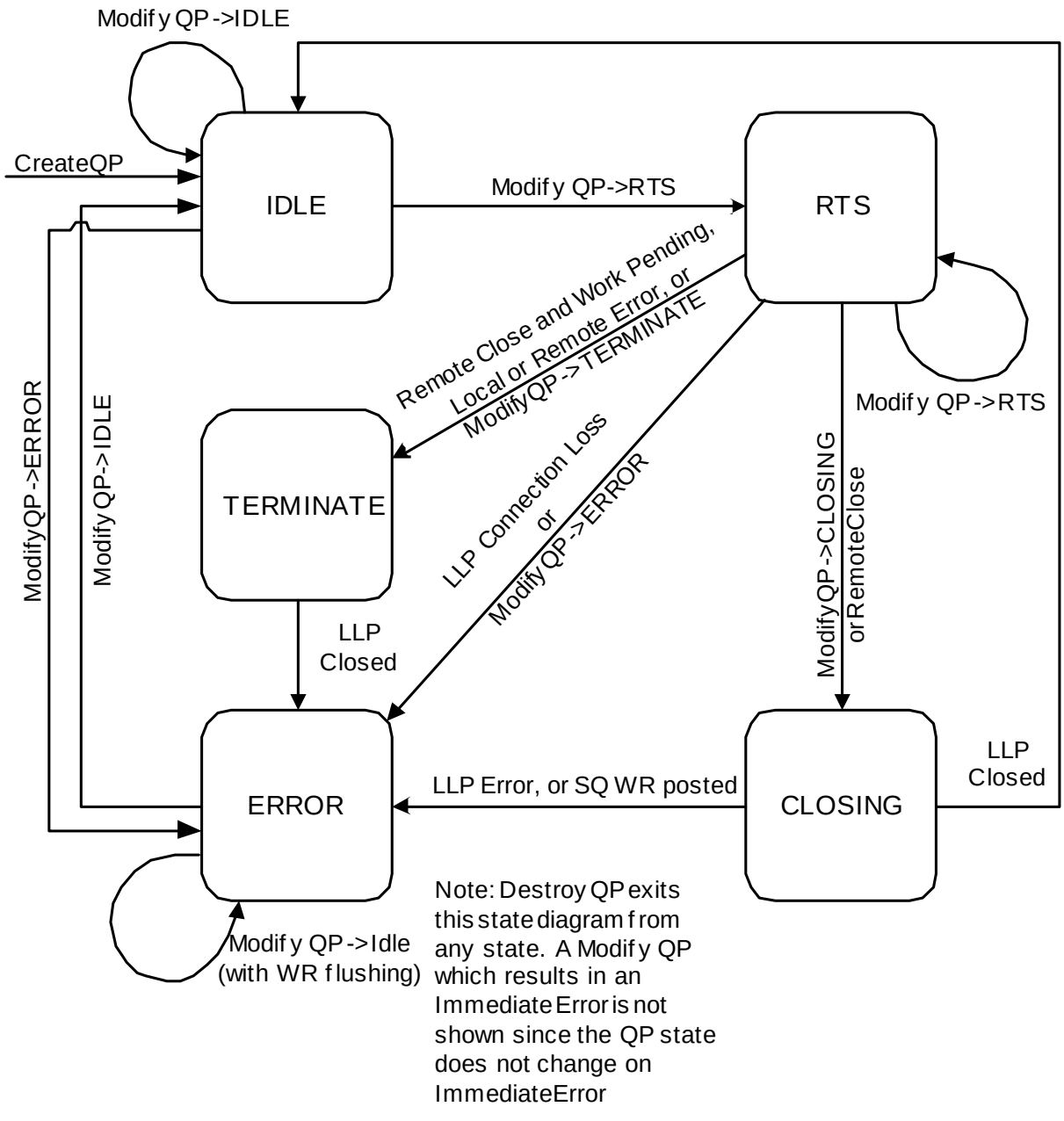


Figure 6 - QP State Diagram

6.2.1 Idle State

The QP MUST be in the Idle state following QP creation or when moved to this state with Modify QP. In this state, Send or Receive WRs MAY be posted but they MUST NOT be processed and CQEs MUST NOT be generated.

Note that whether or not the Consumer posts WRs to the Send Queue when the QP is in the Idle state depends on the method chosen for connection initialization (see Section 6.6.1 - Connection Initialization).

While in the Idle state the RI MUST NOT associate an LLP stream to the QP.

The RI MUST return an Immediate Error if the Consumer attempts to transition the QP from the Idle state to the Terminate state or to the Closing state.

A short summary table describing the state changes for Idle state is shown in Figure 7. The following are detailed descriptions of those changes.

Note that under certain conditions the Consumer might be required to flush Work Requests from a prior RDMAP Stream when in the Idle state. This can be done by transitioning the QP from the Idle to Error state (the Error state flushes all WRs) and then back to the Idle state. This may be necessary if when the Idle state is reached automatically (i.e. no Consumer intervention) from the RTS state at the Local Peer, which will occur if:

* the QP is currently in the RTS state, and the Consumer is actively posting Work Requests (PostSQ or PostRQ),

* the Remote Peer initiates an LLP Close (e.g. for TCP, it generates a FIN segment),

* the Local RI receives the LLP Close request, and immediately transitions to Closing state,

* the RI automatically creates an LLP Close acknowledgement (i.e. for TCP, it generates a FIN ACK segment), thus finishing the LLP Close from the Local Peer’s perspective,

* the RI flushes all WRs, and no errors occurred during the LLP Close or flush,

* the RI automatically transitions the QP to the Idle state,

* the Consumer is not aware of the transition to Idle, and posts a Work Request thinking it can still transmit or receive data.

Note that a normal close should only be done by a ULP after an end-to-end synchronization to ensure all outstanding Work Requests have been flushed end-to-end. This is because RDMAP does not provide a graceful close. Thus if the Consumer performed a PostSQ, it is an error made by the Consumer. However, if the ULP posted an extra PostRQ buffer, it is arguable whether this is an error made by the Consumer or not. In either case, to recover the resources before reusing the QP, the Consumer should cause the QP to transition to Error state to flush the WQEs on the SQ and RQ, and then transition the QP back to the Idle state.

6.2.1.1 Idle to Idle

The Modify QP Verb MUST allow a transition of the QP from the Idle state to the Idle state. This is to allow certain Queue Pair Context attributes to be modified in this state before an association with a Remote Peer's QP has been established.

6.2.1.2 Idle to RTS

The Modify QP Verb MUST allow a transition from the Idle state to the RTS state. This is to support LLP Stream establishment. For this transition, the Modify QP Verb requires an LLP Stream Handle, and allows a Stream Message Buffer as well as other Input Modifiers. In order to transition from Idle to RTS, the LLP must be in its "Established" state, able to send and receive data. If not, the Modify QP Verb MUST return an Immediate Error. For more details on LLP Stream establishment, see Section 6.6.1 - Connection Initialization.

The RI performs the following actions in the Idle to RTS transition, which MAY be performed in order:

1. The RI resets the RDMAP, DDP and MPA layers to the initial conditions specified in the appropriate specifications. For example, the DDP Untagged Message Sequence Numbers (MSN) for the Receive queue & IRRQ, and the MPA marker position must be reset as described in [RDMAP], [DDP], and [MPA].

2. If the Modify QP Verb includes a Stream Message Buffer to send, it is RECOMMENDED that the RI performs the following list in order:

    1. The implementation should stop receiving messages from the LLP Stream.

   2. The RI should transmit the specified message buffer to the Remote Peer in streaming mode.

   3. The RI should associate the LLP Stream with the RDMAP, DDP, and MPA layers and the RI should enable the QP to receive and transmit iWARP messages.

   4. The implementation should resume receiving messages from the LLP Stream.

3. If the Modify QP Verb does not include a Stream Message Buffer to send, the RI should associate the LLP Stream with the RDMA, DDP, and MPA layers and the RI should enable the QP to receive and transmit iWARP messages.

4. The RI moves the QP to the RTS state and begins normal operation.

The RI MAY implement the Verb in other ways, but the end result MUST:

1. Associate RDMAP and Lower layers with the QP;

2. While in streaming mode, transmit any Stream Message Buffer that was included in the Modify QP;

3. Ensure that the QP enables reception and transmission of iWARP messages; and,

4. That regardless of how quickly the remote side returns the first iWARP message, ensure that messages MUST NOT be lost.

For example, if the Verb did not stop the LLP receive side, the following race condition MUST be handled properly:

1. The Associated QP transitions to RTS,

2. It begins transmitting RDMA packets,

3. Then the rapid arrival of an iWARP message from the Remote Peer occurs while the Local Peer is transitioning, but not completed the transition, to the RTS state.

Note that the Modify QP Idle to RTS transition that includes a Stream Message Buffer to send may take a significant amount of time to complete. This is due to the requirement to reliably transmit the stream message.

6.2.1.3 Idle to Error

The Modify QP Verb MUST allow the Consumer to modify the QP from the Idle state to the Error state.

If it becomes necessary to remove WQEs posted to the queues in the Idle state, the Consumer may Modify the QP to the Error state, and then back to Idle. Any WQEs on the SQ & RQ will be Completed with a Flushed status by this procedure. This procedure will not change the Completion Status of CQEs already Completed on the CQ. The Consumer can then Poll for Completion on the Completion Queue and examine the Completion Status to determine which WRs were flushed.

Note there is no effect on the LLP since no LLP Stream has been associated with the QP at this point.

Event	Action	Next State
PostSQ, PostRQ	Enqueue WQE	Idle
WQE is present on or added to the tail of the SQ	WQE is NOT processed	Idle
Modify QP->Idle (Footnote 4)		Idle
Modify QP->RTS and Stream Message Buffer included	Reset RDMAP and Lower layers to their initial conditions. Associate RDMAP and Lower layers with QP. Transmit the specified Stream Message buffer in Streaming mode and enable iWARP mode as described in 6.2.1.2.	RTS
Modify QP->RTS with NO Stream Message Buffer included	Reset RDMAP and lower layers to their initial conditions. Associate RDMAP and lower layers with QP. Enable iWARP mode.	RTS
Modify QP->Error		Error
PostSQ/PostRQ error	Return an Immediate Error	Idle
Modify QP, results in error	Return an Immediate Error	Idle

Figure 7 - Idle State summary

Footnote 4: This transition allows changing QP parameters as defined in Figure 4.

6.2.2 RTS (Ready to Send) State

The RTS state is the main operational state for iWARP operation. All normal message processing, both incoming and outgoing, occurs in this state.

The QP MUST be in the RTS state to begin transmitting and receiving any messages. Prior to moving to this state, the LLP Connection & LLP Stream MUST be fully established.

Once in this state, any WQEs already posted on the Send Queue will begin processing. Any new WQEs posted MUST be added to the tail of the queue, (and begin processing, if the queue is empty). Once in this state, valid incoming iWARP Messages MUST be processed, placed and Completed. In this state, posted Receive WRs will be added to the Receive Queue (or S-RQ), processed when a Send Operation Type arrives, and Completed as described in Section 8.2.4 - Completed Work Requests.

The RTS state MAY be left automatically by any of a variety of processing Errors, which will cause a transition to either the Terminate or Error state. See Section 8.3 - Error Handling for details on which errors result in transitioning to which state.

The RI MUST return an Immediate Error if the Consumer attempts to transition the QP from the RTS state to the Idle state.

A short summary table describing the state changes for RTS state is shown in Figure 8. Following are detailed descriptions of those changes.

6.2.2.1 RTS to RTS

The Modify QP Verb MUST allow the Consumer to modify the QP from the RTS state to the RTS state. This allows certain QP parameters to be changed while the QP is Associated with another QP through an LLP Stream.

Among the parameters that MAY be changed are IRD and ORD, the maximum number of WQEs supported by the SQ or RQ. A Consumer should take care when making changes to these parameters in order to prevent potential race conditions between the Modify operation, the posting of operations on the Send and Receive Queue, and incoming messages. For example, reducing the size of the Send or Receive Queue can only be done when there are fewer WQEs present on the queue than the new size. It is the responsibility of the consumer to track the number of outstanding WR on the SQ and RQ if it intends to modify the size of the SQ or the RQ. For IRD and ORD details, see Section 6.5 - Outstanding RDMA Read Resource Management.

6.2.2.2 RTS to Closing

If the Remote Peer begins an LLP Close operation that does not include a Terminate Message (e.g. for TCP a FIN was received), the RI MUST cause the QP to leave the RTS state automatically. If all Send Queue Work Requests and Remote RDMA Read Operations (i.e. incoming RDMA Read Request Messages and associated RDMA Read Response Messages) are completed, the QP MUST transition to the Closing state; If this is not true, or a Terminate Message was received, the QP MUST transition to the Terminate state (see following section). In all of the above cases the RI MUST create an Affiliated Asynchronous Event to report the transition.

The Modify QP Verb MUST allow the Consumer to modify the QP from the RTS state to the Closing state, to begin an LLP Close operation (e.g. for TCP a FIN segment is generated), and MUST NOT generate an Affiliated Asynchronous Event. See Section 6.6.2.1 - Normal Close for more details. When doing a Modify QP to Closing, all Send Queue Work Requests should have been previously Completed, any Remote RDMA Read Operations should have been previously finished, and the Consumer should have stopped posting PostSQ operations, so that no work remains for the QP to do. If this is not the case, the RI MUST ensure that either of the following actions are taken:

* The Modify QP MAY cause a transition to the Closing state which is immediately followed by a transition to the Error state (due to the SQ being non-empty).

* The Modify QP MAY cause a transition to the Closing state followed by a transition to the Idle state (because the SQ was originally empty, the LLP Close completed, causing the transition to the Idle state, and yet the Consumer was still posting SQ operations).

If this Modify QP Verb completes without error, the QP has successfully transitioned to the Closing state (although it may have already transitioned out of the Closing state).

6.2.2.3 RTS to Terminate

The Modify QP Verb MUST allow the Consumer to modify the QP from the RTS state to the Terminate state. This enables the Consumer to inform the Remote Peer that an Abnormal ULP Termination of the connected stream is being done. The Modify QP will result in the Error Code subfield of the Terminate Control Field of the Terminate Message (See [RDMAP]) having a value of 0x0000: Local Catastrophic Error. The Terminate Buffer will then be available to the Local node via Query QP and to the Remote Peer through Query QP (provided the Terminate Message arrives at and is processed by the Remote Peer).

When this Verb completes, the QP is in the Terminate state. For more details, see 6.6.2.2 - ULP Initiated Termination.

The RTS to Terminate state transition MUST occur automatically following: a locally detected error; a Remote Peer beginning an LLP Close (e.g. for TCP a FIN was received) with either local Send Queue WQEs incomplete, or local Remote RDMA Read Operations incomplete; operation error; or any other error that would cause the RI to generate a Terminate Message. If the transition to the Terminate state is due to other locally detected errors, the RI MUST create the appropriate Asynchronous Error Event reporting that error. See Section 8.3.3 - Asynchronous Errors.

The WR, if any, which caused the QP to enter into the Terminate state MUST be completed with the correct Completion Error Code for the error through the CQ associated with the WQ that experienced the error.

If a remote Terminate Message is received, the Terminate state MUST be automatically entered and an Asynchronous Error Event MUST be reported with a status of "Termination Message Received". In this case, the RI MUST NOT send a Terminate Message back to the Remote Peer. Note that if TCP is the LLP, depending upon implementation of LLP Close, the RI may immediately transition to the Error state or it may wait for a TCP ACK before the transition.

6.2.2.4 RTS to Error

The Modify QP Verb MUST allow the Consumer to modify the QP from the RTS state to the Error state. This enables the Consumer to perform an Abnormal ULP initiated Abortive Teardown (for more details, see Section 6.6.2.3 - ULP Initiated Abortive Teardown).

An LLP failure that prevents further transmissions will also cause the RTS to Error transition.

When the QP transitions from the RTS state to the Error state, the LLP stream MUST NOT be associated with the QP.

The following are done prior to entering Error state:

* The RI MUST stop processing SQ WRs, Remote RDMA Read Operations, and any incoming iWARP Segments targeting the QP. See Section 6.4 - Stopping QP processing and Sending the Terminate Message for additional information.

* If the LLP Stream has not closed, an LLP Reset MUST occur

* The LLP Stream resources MUST no longer be associated with the QP once the LLP actions, if any, are taken.

* If this transition is due to a failure of the LLP, the RI MUST create an Asynchronous Error event reporting the error.

When the prior items complete, the QP MUST be transitioned to the Error state.

Event	Action	Next State
PostSQ, PostRQ	Enqueue WQE	RTS
Valid iWARP Segment Arrives	Process Segment	RTS
WQE is present on or added to Send Queue	Process WQE(s) and send data (as necessary)	RTS
Modify QP->Closing	Begin LLP Graceful Close	Closing
Modify QP->RTS	Modify QP parameters as document in Section 6.5	RTS
Modify QP->Error	Stop QP processing,LLP Reset & LLP Disassociated	Error
Modify QP->Terminate	Generate Terminate Message	Terminate
PostSQ/PostRQ error	Return an Immediate Error	RTS
Modify QP, resulting in an Immediate Error	Return an Immediate Error	RTS
LLP Failure that prevents transmission of the Terminate Message	Stop QP processing, LLP Reset, LLP Disassociated, Create Asynchronous Error	Error
LLP Failure that allows transmission of the Terminate Message	Generate Terminate Message	Terminate
Local incoming RDMA Message processing error (RDMA Read Request, RDMA Read Response, or RDMA Write handling)	Generate Terminate Message	Terminate
Local incoming Send Type Message Processing Error	Generate Terminate Message	Terminate
Local WQ processing error	Complete WR as necessary, Generate Terminate Message	Terminate
Received Terminate Message		Terminate
LLP Close Received AND (SQ NOT empty OR IRRQ NOT empty)	Generate Terminate Message	Terminate
LLP Close Received AND SQ empty AND IRRQ empty		Closing

Figure 8 - RTS State summary

6.2.3 Terminate State

The Terminate state is used to send the final Terminate Message and begin an LLP Close if an error has occurred, or as a staging ground to perform an LLP Close if a Terminate Message was received from the Remote Peer. This state is transitory. The duration is limited by the time to finish the LLP Close operation or a final timeout in LLP Close (which would cause an LLP Reset).

When the Terminate state is exited to the Error state, the LLP Stream MUST no longer be associated with the QP and the LLP Stream MUST be in either a condition of LLP Closed or LLP Reset.

It is possible to examine the Terminate Message buffer while in this state by using Query QP (Section 9.2.5.2) to retrieve the Terminate Message.

A short summary table describing the state changes for the Terminate state is shown in Figure 9. The following are detailed descriptions of those changes.

While in the Terminate state, the following are done:

* The RI MUST stop processing SQ WRs, Remote RDMA Read Operations and any new incoming iWARP Segments targeting the QP. For additional information, see Section 6.4 - Stopping QP processing and Sending the Terminate Message.

* The RNIC MUST attempt to send the RDMAP Terminate Message, indicating the cause of error, except when the Terminate state is entered due to reception of a remote Terminate Message. Note that sending the Terminate Message may not be successful if an LLP Reset occurs.

* The RI MUST begin an LLP Close operation.

* If the current stream is the last (or only) active LLP Stream on the LLP Connection, or the LLP is in a state where all streams are unable to operate, the LLP Close MUST cause the LLP Connection to be closed. (For example, in [TCP] the FIN is sent and the close sequence is done.)

* If an LLP error occurs during the sending of the Terminate Message (including reception of an incoming LLP Reset, between the time the Terminate state is entered and the LLP Close sequence is completed), or due to an LLP final timeout while the LLP Close operation is not finished, then an LLP Reset MUST occur and its resources MUST no longer be associated with the QP. Note that the LLP MUST use a timeout to detect errors, so that the QP is in the Terminate state for a bounded time.

* At some point in the Terminate state, the RI MUST begin to return an Immediate Error for any attempt to post a WR to a Work Queue; prior to that point, WQEs MUST be enqueued (and eventually flushed) or result in an Immediate Error.

* The RI MAY begin to flush any incomplete WRs on the SQ or RQ. Please see the Section 6.2.4 - Error State for further requirements about flushing incomplete WRs.

* When the prior actions are done:

    1. If the transition to the Terminate state is due to the Modify QP Verb, the RI MUST NOT create an Asynchronous Error Event reporting "Error State Entered". If the transition to the Terminate state is due to the Modify QP Verb, but an LLP error occurred while in the Terminate state, then the RI MUST generate an Asynchronous Error reporting "Bad Close".

    2. If the transition to the Terminate state is due to an error that is reported in a Work Completion, the RI MUST NOT create an Asynchronous Error. See Section 8.3.2 - Work Completion Errors. If the transition to the Terminate state is due to an error that is reported in a Completion, but an LLP error occurred while in the Terminate state, then the RI MUST generate an Asynchronous Error reporting "Bad Close".

When the actions listed above are complete, and the LLP Close is finished, the QP state MUST move automatically to the Error state.

When the LLP Close is finished or an LLP Reset occurs, the RI MUST disassociate the QP from the LLP Stream, including any LLP Stream context and any resources associated with it. Disassociating the LLP Stream from the QP means that it becomes possible for the QP to be transitioned to Idle and to RTS with a new LLP Stream.

Any attempt to perform a Modify QP in the Terminate state MUST return with an Immediate Error.

Event	Action	Next State
On entry	Stop QP processing, Send & attempt to complete Terminate Message if one wasn't received. LLP Close Initiated	Terminate
LLP Close complete	Create Asynchronous Event if necessary, LLP Disassociated from QP	Error
LLP Failure that prevents transmission of the Terminate Message	LLP Reset and create Asynchronous Event if necessary, LLP Disassociated from QP	Error
Valid IWARP Segment Arrives	Ignore Segment	Terminate
PostSQ/PostRQ error	Return an Immediate Error	Terminate
Modify QP	Return an Immediate Error	Terminate
WQE is present on or added to a Work Queue	WQE is NOT processed and is eventually flushed.	Terminate

Figure 9 - Terminate State summary

6.2.4 Error State

The Error state provides an indication that the QP has experienced an error (or transitioned to the Error state through the use of a Modify QP) and has stopped operations. On entry to the Error state, the LLP Stream MUST NOT be associated with the QP.

The RI MUST return an Immediate Error if the Consumer attempts to transition the QP from the Error state to the RTS, Terminate, or Closing state.

The following is done on entry into the Error state:

* The RI MUST flush any incomplete WRs on the SQ or RQ. All WQEs on the SQ and RQ, except for the WQE that caused the error (if any), MUST be returned with the Flushed Error Completion Status through the Completion Queue associated with the WQ. Note that the WQE which caused the error may not be at the head of the Work Queue. The Consumer should expect in some cases to retrieve Work Completions with the Flushed Error Completion Status, as well as potential successful completions, before retrieving the WC for the WR which caused the error. The RI MUST NOT return more than one Work Completion with a Work Completion Status set to something other than the Flushed Completion Status or the Success Completion Status.

* At some point in the execution of the flushing operation, the RI MUST begin to return an Immediate Error for any attempt to post a WR to a Work Queue; prior to that point, any WQEs posted to a Work Queue MUST be enqueued and then flushed as described above (e.g. The PostSQ is done in Non-Privileged Mode and the Non-Privileged Mode portion of the RI has not yet been informed that the QP is in the Error state).

If a Terminate Message was sent or received, the RI MUST allow the Consumer to retrieve it through the Query QP Verb (Section 9.2.5.2).

Following entry to the Error state, and before Destroying the QP or restarting the QP by going through Idle to RTS, it may be necessary to clean up some of the resources associated with the QP.

* Work Completions should be reaped by using Poll for Completion (Poll CQ) (see Section 9.3.2.1) before destroying the QP, otherwise they may become inaccessible.

* Memory Window resources MUST be deallocated by using Deallocate STag (see Section 9.2.6.4). This is necessary since in the Valid state they are associated with the QP. QP destruction will fail when Memory Windows which are in the Valid state are still Bound to the QP.

* Memory Regions can be invalidated by posting an Invalidate Local STag WR to other SQs in the same PD, or they can be deallocated by using Deallocate STag. If left in the Valid state, the associated memory may be at risk of unexpected remote access.

If the QP is transitioning to the Error state, or has not yet finished flushing the Work Queues, a Modify QP request to transition to the IDLE state MUST fail with an Immediate Error. If none of the prior conditions are true, a Modify QP to the Idle state MUST take the QP to the Idle state. No other state transitions out of Error are supported. Any attempt to transition the QP to a state other than Idle MUST result in an Immediate Error.

A short summary table describing the state changes for Error state is shown in Figure 10.

Event	Action	Next State
On Entry	Flush any incomplete WQEs	
Modify QP->Idle (no outstanding WRs and not in transition to Error)		Idle
Modify QP->Idle (outstanding WRs or in transition to Error)	Return an Immediate Error	Error
Post WR	Post WQE, and then Flush it, OR Return an Immediate Error	Error
Modify QP, resulting in an error	Return an Immediate Error	Error

Figure 10 - Error State summary

6.2.5 Closing State

This state is used to wait for the LLP to complete the LLP Close, if no errors occurred. For some LLPs or some RI implementations, moving a QP from the RTS state to the Idle state can require an end-to-end acknowledgement or require the Remote Peer to close their half of the LLP Stream before the LLP Close is finished. This may take a significant amount of time. Thus the Closing state is provided so that these operations are done in a fashion that is visible to the Consumer. Note that some RI implementations may require the LLP Stream to be completely closed before transitioning to the Idle state. This can be in the order of tens of seconds (e.g. an RI implementation on TCP may require TCP to be in the CLOSED state, possibly waiting in the TIME-WAIT state for a significant amount of time).

If the LLP Close operation does not require the LLP to transmit messages (e.g. for SCTP there is no mechanism to close a single LLP Stream, thus when one LLP Stream is closed and other LLP Streams remain active, there is no end-to-end handshake required), then the RI MAY transition rapidly through this state.

When the Closing state is exited to Idle, the LLP Stream MUST NOT be associated with the QP.

Any attempt to perform a Modify QP in the Closing state MUST return an Immediate Error.

Errors detected by the RI when the QP is in the Closing state result in a transition to the Error state; for LLP failures, this is indicated with the specific Asynchronous Event "LLP Connection Lost".

A short summary table describing the state changes for the Closing state is shown in Figure 11. Following are detailed descriptions of those changes.

The following are done prior to exiting Closing state:

* The RI MUST stop processing SQ WRs and Remote RDMA Read Operations targeting the QP.

* The RI MUST stop processing any incoming segments, though the RI MAY process any arriving Terminate Messages.

* At some point in the Closing state the RI MUST begin to return an Immediate Error for any attempt to post a WR to a Work Queue; prior to that point, WQEs MUST be enqueued or result in an Immediate Error.

* The RI MUST flush all incomplete WQEs on the RQ. All WQEs on the RQ MUST be returned with the Flushed Error Completion Status through the Completion Queue associated with the RQ. If RQ WQEs are enqueued, the RI MUST flush the WQE with the Flushed Error Completion Status through the Completion Queue associated with the RQ.

* If no errors have been detected (see next bullet), an LLP Close MUST occur. If the LLP Stream is the last or only active stream for the LLP Connection, the LLP Connection MUST be attempted to be closed gracefully. (For example, in [TCP] the FIN is sent and close sequence is done.).

* The RI MUST generate an Asynchronous Error if:

  - Any SQ WQEs were on the SQ at any time during the Closing state. Note, this condition may happen if the PostSQ is done in Non-Privileged Mode and the Non-Privileged Mode portion of the RI has not yet been informed that the QP is in the Closing state. Also, the Error state will flush all SQ WQEs.

  - Any incoming data arrives during the LLP Close. If the incoming data is a Terminate Message, the RI MAY allow the Consumer to retrieve the Terminate Message through the Query QP Verb.

  - Any Remote RDMA Read Operations are in process.

  - An LLP Stream failure (e.g. LLP Stream is lost) occurs during the LLP Close. Note that the RI MUST use a timeout mechanism to detect LLP errors during the LLP Close, so that the QP is in the Closing state for a bounded time. If the LLP detects a final timeout, it MUST be considered an error.

* If the RI generates an Asynchronous Error, the following MUST occur in order:

  - An LLP Reset MUST occur and the LLP resources MUST no longer be associated with the QP.

  - The QP MUST be transitioned to the Error state.

  - The RI MUST generate an Asynchronous Event

* If no error occurs during the LLP Close operation:

  - When all RQ WRs have been flushed and the LLP Close has finished, the LLP Stream MUST be disassociated with the QP, the RI MUST generate an Asynchronous Event "LLP Close Complete".

- When the prior items complete, the QP MUST be transitioned to the Idle state.

When the LLP Close is finished or an LLP Reset occurs, the RI MUST disassociate the QP from the LLP Stream, including any LLP Stream context and any resources associated with it. Disassociating the LLP Stream from the QP means that it becomes possible for the QP to be transitioned to Idle and to RTS with a new LLP Stream.

Note that it is possible for the Consumer to post WRs while the automatic transition from RTS to Closing to Idle is occurring. See Section 6.2.1 - Idle State for additional details.

Event	Action	Next State
On Entry	Stop QP processing, start LLP Close, and start Flushing any incomplete WQEs on Receive queues.	Closing
LLP Close complete, all RQ WQEs flushed, and no SQ WQEs on the SQ	Create Asynchronous Event: "LLP Close Complete", LLP Disassociated from QP	Idle
At least one SQ WQE on the SQ or Remote RDMA Read Operation in progress.	Perform LLP Reset, Create Asynchronous Event: "Bad Close", LLP Disassociated from QP	Error
LLP Connection Failure	Perform LLP Reset, Create Asynchronous Event: "LLP Connection Lost", LLP Disassociated from QP	Error
Segment Arrives and is not a Terminate Message	Perform LLP Reset, Segment is not processed. Create Asynchronous Event: "Bad Close", LLP Disassociated from QP	Error
Segment Arrives and is a Terminate Message	Perform LLP Reset, MAY create Async Event: "Bad Close"; MAY allow examination of Terminate Message, LLP Disassociated from QP	Error
PostSQ/PostRQ with Immediate Error	Return an Immediate Error	Closing
Modify QP	Return an Immediate Error	Closing
PostRQ without Immediate Error	Enqueue and flush	Closing
PostSQ without Immediate Error	Enqueue & Flush, Perform LLP Reset, Create Async Event "Bad Close", LLP Disassociated from QP.	Error

Figure 11 - Closing State summary

6.3 Shared Receive Queue

The Verbs support a Shared Receive Queue (S-RQ). Support for the Shared Receive Queue is OPTIONAL. The Query RNIC Verb MUST indicate whether the RNIC supports the Shared Receive Queue.

A Shared Receive Queue is an RNIC resource which allows multiple RQs to retrieve WQEs from the same shared queue on an as needed basis. This allows a Consumer to post WRs to the S-RQ instead of the RQ. When a message arrives, the RI uses a WQE from the S-RQ and makes it appear as if the WQE has been copied from the S-RQ to the QP's RQ. A CQE for an incoming message which result in a WQE being consumed from an S-RQ MUST be posted to the CQ associated with the QP's RQ.

The RI MUST return the maximum number of S-RQs supported by the RI as an output modifier of Query RNIC, and the value MUST be zero if the RI does not support S-RQs.

The RI MUST return the maximum number of outstanding WRs on an S-RQ as an output modifier of Query RNIC, and the value MUST be zero if the RI does not support S-RQs.

Each S-RQ MUST be associated with a single PD ID. Multiple S-RQs MUST be able to be associated with the same PD ID.

The SQ of a QP associated with an S-RQ MUST operate no differently than the SQ of a QP which is not associated with an S-RQ.

When using an S-RQ, the RI MUST allow Work Requests to be posted to the S-RQ and MUST NOT allow WRs to be posted to an RQ of a QP associated with the S-RQ.

If the RI supports an S-RQ, then it MUST:

* support the Create S-RQ Verb (See Section 9.2.4.1),
* support the Query S-RQ Verb (See Section 9.2.4.2),
* support the Modify S-RQ Verb (See Section 9.2.4.3),
* support the Destroy S-RQ Verb (See Section 9.2.4.4),
* support the S-RQ Handle as an Input Modifier for Create QP (See Section 9.2.5.1), and
* support an S-RQ Limit Event and a QP RQ Limit Event (See Section 6.3.8),
* support the S-RQ Handle as an Input Modifier for PostRQ,

* support the S-RQ Handle as an Asynchronous Event Handler routine parameter.

6.3.1 Creating a Shared Receive Queue

When the S-RQ is created, it MUST be associated with a PD ID, and the maximum number of WRs which can be posted at any time must be provided as an Input Modifier. Note that the number of WQEs on the S-RQ at any given moment is dependent upon the completion semantics described below.

6.3.2 Modifying a Shared Receive Queue

The RI MAY allow the Consumer to change the maximum number of outstanding WRs on the S-RQ. If the RI supports the ability to change the number of outstanding WRs on a SQ and RQ, and the RI supports S-RQs, then it MUST:

* allow the maximum number of outstanding WRs on the S-RQ to be changed;

* allow the maximum number of outstanding WRs to be changed while WRs are still outstanding; and

* support the ability to change this on every S-RQ.

It is understood that changing the number of WRs that an S-RQ may have outstanding MAY adversely affect performance. Resizing the S-RQ MUST NOT cause Immediate, Completion or Asynchronous Errors, with the exception of Immediate Errors returned by the Modify S-RQ Verb and possible LLP time-outs. It is expected that the resize operation MAY adversely affect the Associated QPs attempting to communicate with the QPs associated with the S-RQ during the resize operation possibly resulting in LLP time-outs and retries which could result in LLP Stream teardown (which would result in an Asynchronous Error). It is suggested that the Consumer only perform this resize operation when activity on the connections has been quiesced to minimize the risk of transitioning Associated QPs to the Error state as a result of LLP time-outs.

If the number of requested outstanding WRs is smaller than the actual number of outstanding WRs currently on the S-RQ, then the modification of the S-RQ MUST fail with an Immediate Error and the S-RQ MUST remain in the original state.

6.3.3 Destroying a Shared Receive Queue

The Verbs provide a Destroy S-RQ Verb to allow a Consumer to destroy an S-RQ that is no longer needed. The RI MUST only allow an S-RQ to be destroyed when all the QPs associated with that S-RQ have been

destroyed. The RI MUST allow an S-RQ to be destroyed when there are WRs still posted to the S-RQ. Note that it is recommended that a Consumer drain the S-RQ or track all WRs posted to the S-RQ before destroying it so that no WRs are lost. For example, a WR which was Posted to the S-RQ but which was never Completed would still be on the S-RQ when the S-RQ was destroyed so the Consumer would never be notified that the buffers associated with the WR were available again.

After the Destroy S-RQ returns to the Consumer, the RI:

* MUST have freed all RI resources associated with Receive Work Requests that were not Completed and were posted on that S-RQ, and

* MUST ensure that it will no longer reference any Consumer resources associated with Receive Work Requests that were not Completed and were posted on that S-RQ.

6.3.4 Associating an S-RQ with a QP

A Shared Receive Queue MUST only be associated with a QP when the QP is created. When the QP is created, the RI MUST ignore the maximum number of outstanding RQ WRs Input Modifier.

6.3.5 Shared Receive Queue Processing Model

If a QP is associated with an S-RQ, the RI MUST allow WRs to be posted to the S-RQ using PostRQ, specifying the S-RQ Handle instead of the QP Handle. If the QP is associated with an S-RQ, the RI MUST NOT allow WRs to be posted to the Local RQ through PostRQ and MUST return an Immediate Error if Posting to the Local RQ is attempted by the Consumer.

The RI MUST ensure that S-RQs follow the rules for Work Queues with respect to the posting rules and completion rules defined in Section 8.2.1 - Submitting Work Request to a Work Queue and Section 8.2.3 - Completion Processing. This means the RI MUST prevent a Consumer from overflowing the S-RQ using the PostRQ.

When an incoming Untagged Message arrives on a QP, the RI determines if the QP is associated with an S-RQ. If it is, the RI must make it appear as if the WQE has been dequeued from the S-RQ and queued to the QP's local RQ. This does not guarantee that the S-RQ WQE is free. The S-RQ WQE is considered to be part of the S-RQ until the Work Completion associated with the S-RQ WQE has been retrieved or the S-RQ is destroyed.

The RI MAY dequeue or use the S-RQ WQEs in any order. Since the WQEs are in an implementation specific order, the Consumer should not

depend on S-RQ post order in any way. The RI should support one of the following two models: sequential order or arrival order.

* In sequential ordering, the RI dequeues S-RQ WQEs as messages arrive. If messages arrive out of order, in addition to dequeueing the WQE required to place the data for that message, the RI also dequeues a WQE for each message with an MSN lower than the out-of-order message that has not arrived and does not yet have an associated WQE.

* In arrival ordering, the RI dequeues S-RQ WQEs as the messages arrive. If messages arrive out of order, only the WQE required to place the out of order message will be dequeued from the S-RQ. WQEs required to place data for the messages with an MSN lower than the out of order message will be dequeued from the S-RQ when those messages arrive.

The RI MUST Complete incoming Send Message Types in the order they were Posted to the Associated QP's Send Queue. This means Work Completions retrieved from the CQ for any individual QP will be retrieved only in Message Sequence Number (MSN) order (see [DDP] for details). The RI MUST dequeue only one WQE from the S-RQ to place any message represented by a single MSN. Note that the Work Completions are not necessarily in the order in which the Send Message Types arrived, nor in the order the WQEs were posted to the S-RQ, nor in the order the WQEs were dequeued from the S-RQ.

When a Work Completion which represents a WR originally submitted to an S-RQ has been returned to the Consumer via the Poll for Completion Verb, the RI MUST allow the Consumer to be able to post another Work Request to the S-RQ immediately.

All QPs that use an S-RQ MUST be able to consume S-RQ WQEs, as long as the S-RQ has unconsumed WQEs available. If there are no S-RQ WQEs when an Untagged Message arrives on a QP which is associated with that S-RQ, then the LLP Stream MAY be Terminated. If the LLP Stream is not terminated, the reader should see Section 13.2 - Graceful Receive Overflow Handling for one implementation option.

Protection Domain checking rules are slightly different for an S-RQ. An S-RQ MUST have a PD ID assigned as an Input Modifier for Create S-RQ. When an Untagged Message arrives and the QP has been determined to use an S-RQ for its incoming Untagged Message WQEs, then the PD ID of the STags in the WQEs MUST be validated against the PD ID of the S-RQ and MUST NOT be validated against the PD ID of the QP.

Note that due to the Protection Domain checking rule above, the Consumer will not be able to invalidate an STag used by the S-RQ unless the S-RQ's PD is the same as the QP's PD, even if the QP uses

the S-RQ. This is because the PD used for comparison in Invalidation operations is that of the QP, not the S-RQ.

The use of the STag of zero as part of a SGE in a WR MUST be validated by the RI based on the QP's attribute which indicates if it is allowed on the QP. If use of the STag of zero is not permitted on the QP and a WQE referencing STag zero is processed on the QP, the RI MUST return a Completion Error. Consequently, if the Consumer uses the STag of zero in S-RQ Work Requests and the S-RQ is accessed by QPs that have the use of STag of zero enabled as well as QPs that do not have the use of STag of zero enabled, then the QPs that do not have the use of STag of zero enabled will transition to the Error state as soon as they retrieve a WQE which contains an STag of zero.

6.3.6 S-RQ Error Semantics

All errors encountered MUST be reported through Work Completions where possible. This is due to the semantic requiring the WQE to appear as if it had been on the QP's RQ. The exception is that a catastrophic S-RQ error MUST be reported as an Affiliated Asynchronous Error.

Errors related to a connection for a QP associated with an S-RQ MUST NOT affect the S-RQ. Any WQEs already consumed by the QP from the S-RQ will be completed in error or flushed in the case of an LLP Stream error. Any other QPs associated with the S-RQ MUST remain unaffected by a local QP error.

Errors related to a Work Request on an S-RQ will be posted to the CQ associated with the QP's RQ if they are processing errors, or returned as Verb results if they are Immediate Errors.

In the case of a catastrophic S-RQ failure, any QP associated with the S-RQ will transition to the Terminate state when the QP attempts to dequeue a WQE from the S-RQ when handling an incoming Send Type Message. The resource ID returned by the Asynchronous Event Handler MUST be the QP ID. All outstanding WQEs on the QP will be flushed and an Affiliated Asynchronous Event: "S-RQ error on a QP" MUST be generated as part of the Terminate state transition.

The RI MUST NOT flush the WQEs on an S-RQ which have not been used to Place incoming Untagged Messages when any associated QP transitions to the Terminate, Error or Closing states.

6.3.7 S-RQ Resource Sizing

The Consumer is responsible for sizing the S-RQ and the CQs associated with the QP's RQs appropriately. The RI MUST ignore the sizing information provided for the QP's RQ when the QP uses an S-

RQ. The Consumer should note this fact when invoking the Create QP Verb using an S-RQ handle. In addition, S-RQs are subject to the Completion confirmation rules defined in Section 8.2.3 - Completion Processing. This means that the WR MUST be considered to be in the scope of the RI, and thus using a WQE on the S-RQ until the Work Completion has been retrieved. In addition, the RI MUST allow any single RQ to utilize all of the WQEs posted to an S-RQ . Note also that the RI is not required to perform CQ overflow detection.

The RQ size Input Modifier is not used when a QP is associated with an S-RQ. In this case, the RQ has no defined size. It can be up to the size of the S-RQ. If the S-RQ is resized, any QP MUST be able to utilize all of the WQEs posted to the S-RQ. It is up to the implementation to process multiple messages in progress at one time. Note that the number of messages that can be in progress at once is limited by the S-RQ size, the LLP receive window, and possibly other factors.

6.3.8 S-RQ Limit Checking

An RI that supports the S-RQ MUST support an S-RQ Limit Notification. An RI that supports S-RQ MUST support an S-RQ Limit input modifier on the Create S-RQ and Modify S-RQ Verbs to establish the value of the Limit. The S-RQ Limit detection MUST be armed by the RI upon creation of the S-RQ, if non-zero. This is only used for generation of the Affiliated Asynchronous Event and MUST NOT otherwise disrupt the QP operation. When the number of available (or unused) WQEs posted to the S-RQ drops below the S-RQ Limit, the RI MUST generate an Asynchronous Event and provide the S-RQ Handle as the Resource ID. This event will only be triggered once after it is armed and will not generate another event until the Consumer re-arms the event. The RI MUST allow the Consumer to re-arm this event through the use of Modify S-RQ. The RI MUST arm this event when the S-RQ is created if the S-RQ Limit is greater than zero. The RI MUST allow an already armed S-RQ Limit to be armed again. If the S-RQ Limit is armed for an S-RQ and the maximum number of outstanding WRs on the S-RQ is modified below S-RQ Limit, then the RI MUST return an Immediate Error indicating that an invalid Input Modifier was provided.

An RI that supports the S-RQ MUST support a QP RQ Limit Notification for QPs associated with an S-RQ. The QP RQ Limit detection MUST be armed by the RI upon creation of the QP, if non-zero. The Consumer specifies the QP RQ Limit as part of either Create QP or Modify QP. This is only used for generation of the Affiliated Asynchronous Event and MUST NOT otherwise disrupt the QP operation. When the number of messages in progress on the QP (which is defined as messages being Placed, and thus have WQEs associated with them, but which have not yet had CQEs generated for the WQEs and thus have not been Delivered to the Consumer) exceeds the QP's RQ Limit, the RI

MUST generate an Asynchronous Event and provide the QP ID as the Resource ID. This event will only be triggered once after it is armed and will not generate another event until the Consumer re-arms the event. The RI MUST allow the Consumer to re-arm this event through the use of Modify QP. The RI MUST arm this event when the QP is created if the QP's RQ Limit is greater than zero. The RI MUST allow an already armed S-RQ Limit to be armed again. If the S-RQ Limit specified in the Create S-RQ or Modify S-RQ is greater than the maximum number of outstanding WRs on the S-RQ, then the RI MUST return an Immediate Error indicating that an invalid Input Modifier was provided.

Note that neither Limit Notification forces Work Completions to be retrieved by the Consumer. Only retrieving the Work Completions allows the Consumer to Post additional WQEs to the S-RQ. Consequently, if separate Consumers are allowed to share an S-RQ, then one Consumer could consume all or part of the S-RQ entries if it does not retrieve Work Completions.

6.4 Stopping QP processing and Sending the Terminate Message

Certain conditions require that QP operations be stopped, and a final Terminate Message be sent. Stopping WR processing on the QP and transmission of a Terminate Message are associated with QP state changes; the specific QP state transitions that require this are described in Section 6.2 - Queue Pair Resource States. When a QP must be stopped, either by a Modify QP Verb, or by QP state change due to an error, the following notes apply:

1. For Errors that do not impact the integrity of an outbound DDP Segment or for Modify QP Verb invocations that require stopping the QP, outbound processing MUST be stopped only on DDP Segment boundaries, in the absence of LLP errors. Any Terminate Message (if required) MUST be filled out as described in [RDMAP] and MUST be sent after the last complete outbound DDP Segment.

   For Errors that impact the integrity of an outbound DDP Segment that require stopping the QP:

   - If the RI has not begun sending the DDP Segment, then outbound processing MUST be stopped before the DDP Segment is sent; and the Terminate Message and error code MUST be sent instead of the erroneous DDP Segment.

   - If the RI has begun sending the DDP Segment, then outbound processing MUST be stopped immediately on the byte that experienced the error and the LLP Stream MUST be Reset.

2. For Errors or Modify QP Verbs (except for RTS to Closing transitions) that require stopping the QP, the RI MUST cease to

process inbound DDP Segments, at least by the time that any currently in-process DDP Segment has completed processing.

The semantics of stopping QP processing and handling incoming DDP segments for Modify QP Verbs that require the transition from RTS to Closing are discussed at length in Section 6.2.5.

Subsequent inbound DDP Segments (if any) are ignored and any inbound DDP Segments that have been Placed but not Delivered are never Delivered.

3. For Modify QP Verbs that require stopping the QP, the RI SHOULD stop outbound QP processing prior to sending any current DDP Segment to the LLP and MUST stop outbound QP processing at least by the time that any currently in-process outbound message has completed processing.

4. For Errors detected while creating RDMA Write, Send Type, or RDMA Read Type Work Requests, the RI MUST stop outbound QP processing prior to sending the current DDP Segment to the LLP. The Terminate Message and Error code MUST be sent instead of the original message (or DDP Segment). In this case, the [RDMAP] Terminate Message's Terminate Control Field is set to represent RDMA and the Error Type is set to represent Local Catastrophic Error.

5. For Errors detected while creating RDMA Read Responses to a Remote RDMA Read Operation, the RI MUST stop outbound QP processing prior to sending the erroneous DDP Segment to the LLP. The Terminate Message and Error code are sent instead of the erroneous RDMA Read Response Message.

6. For Errors detected while creating CQEs, or other reasons not directly associated with creating an outbound DDP Segment, the RI SHOULD stop outbound QP processing prior to sending any current DDP Segment to the LLP and MUST stop outbound QP processing at least by the time that any currently in-process outbound DDP message has completed processing. In this case, [RDMAP] Terminate Message's Terminate Control Field's Header Control Bits are all zero.

7. If an error is detected by an iWARP implementation while an incoming DDP Segment data is being Placed, the error actions (changing state, stopping the QP, etc.) MUST be delayed until after the segment is actually delivered by the LLP. If more than one error is detected on incoming segments, then the first DDP Segment Delivered with a detected error MUST result in the error actions. The first detected error MAY have been detected by the LLP, DDP Layer, or RDMA Layer. If, while waiting for Delivery of an incoming segment that contains an error, another error is

detected that is not associated with incoming segments (for example, an LLP error, Send Queue or RDMA Read Response processing error), then the RI MUST perform the actions for that error without waiting for Delivery of any other segments.

8. For errors detected on incoming DDP Segments (after they have been Delivered by the LLP), the Terminate Message MUST include a copy of the iWARP header from the DDP Segment in error (see [RDMA]).

Below, in Figure 12, is a table which should indicate the values for the fields in the Terminate Control Field of the Terminate Message in [RDMAP].

	Layer	**EType**	**Error Code**	**HdrCt**	**DDP Seg. Lgt**	**Term DDP Hdr.**	**Term RDMA Hdr.**
For Modify QP from RTS to Error	No Terminate Message is sent.						
For Modify QP from RTS to Terminate	RDMA (0x)	Local Catast. (0x)	None (0x)	000b	All zeros	All zeros	All zeros
For Errors detected while creating RDMA Write, Send Type, or RDMA Read Request Messages	RDMA (0x)	Local Catast. (0x)	None (0x)	000b	All zeros	All zeros	All zeros
For Errors detected while creating completions, or other reasons not directly associated with creating an outbound DDP Segment	RDMA (0x)	Local Catast. (0x)	None (0x)	000b	All zeros	All zeros	All zeros
For Errors detected processing or Placing incoming Send Type, RDMA Write, RDMA Read Request or RDMA Read Response Messages	Depends on error, see [RDMAP] specification and/or Sections 8.3.2 & 8.3.3.						

	Layer	EType	Error Code	HdrCt	DDP Seg. Lgt	Term DDP Hdr.	Term RDMA Hdr.
For Errors detected while creating RDMA Read Response Messages	Depends on error, see [RDMAP] specification and/or Sections 8.3.2 & 8.3.3.						
For LLP layer errors detected by an iWARP implementation (e.g. incoming LLP Reset while QP in RTS)	No Terminate Message is sent.						
Incoming Terminate Msg	No Terminate Message is sent.						

Figure 12- Terminate Control Field Values

6.5 Outstanding RDMA Read Resource Management

RDMA allows multiple RDMA Read Request Messages to be outstanding on a single LLP Stream. To enable this feature, the RNIC provides resources associated with both the inbound and outbound stream. For each outbound RDMA Read Request Message, the RNIC has some resources to track the request until a local Completion occurs. Similarly, for each inbound RDMA Read Request Message, the RNIC has an Inbound RDMA Read Request Queue (IRRQ) (associated with the DDP Queue Number of 1) to store the state of the request until it has been satisfied by sending all of the requested data in the RDMA Read Response Message. The Input Modifier that specifies this value is called the Inbound RDMA Read Queue Depth (IRD).

The Outbound RDMA Read Queue Depth (ORD) is the allocated number of outstanding RDMA Read Request Messages the RNIC is allowed to have outstanding at the Data Sink of an RDMA Read Operation. This is the resource used to track the request until a local Completion occurs.

The Inbound RDMA Read Queue Depth (IRD) is the allocated number of incoming RDMA Read Request Messages a QP can support at the Data Source for an RDMA Read Operation. This is the resource used to track inbound RDMA Read Request Messages.

An RNIC MUST implement these resources as either per QP resources, or shared per RNIC resources. Per QP means that the resources are tied to the QP and are most likely part of the QP Context. Per RNIC resources implies that the RNIC has a pool of such resources internally that it assigns to the QP based on the values of IRD and ORD associated with the QP.

Query RNIC MUST return the type of resources the specified RNIC supports. The results are returned in the following Output Modifiers for Query RNIC:

* The maximum number of Inbound RDMA Read Request Queue messages that can be outstanding per RNIC. This is the per RNIC parameter that corresponds to IRD. This value is Zero if the resources for handling Inbound RDMA Read Requests are not shared between QPs.

* The maximum number of Outbound RDMA Read Request messages that can be outstanding per RNIC. This is the per RNIC parameter that corresponds to ORD. This value is Zero if the outstanding RDMA Read Requests are not shared between QPs.

* The maximum number of inbound RDMA Read Request Messages that the Inbound RDMA Read Request Queue can store per QP (corresponds to IRD).

* The maximum number of outbound RDMA Read Request Messages that can be outstanding per QP (corresponds to ORD).

The Consumer is responsible for setting the RDMA Read Data Sink QP's ORD so that it does not exceed the Associated QP's IRD at the Data Source.

If the Consumer attempts to set IRD or ORD to one or greater, and there are not enough resources to allow this, the Create QP or Modify QP Verb MUST fail with an Immediate Error. This can happen because the maximum amount of IRD/ORD resources returned by Query RNIC MAY be affected by consumption of unrelated resources, so that not all of the reported resources may actually be available simultaneously.

If the IRD and ORD resources are not shared between QPs (e.g. fixed per QP instead of allocated out of a pool for the RNIC), then the ULP need only negotiate the values for IRD and ORD. But if the IRD and ORD resources are shared across the RNIC, then some function of the Consumer or Consumer's environment (such as a resource manager) must determine how to allocate the resources among the QPs in addition to negotiating the IRD and ORD values.

The RNIC MUST ensure that it does not issue more RDMA Read Request Messages than is specified by the QP's ORD value. However, the RI MUST allow the Consumer to post as many RDMA Read Type Work Requests as it can, within the limit of the total Work Requests the Send Queue can support. The RI MUST delay processing of an RDMA Read Type Work Request posted to the SQ which would result in exceeding the QP's ORD value until a prior RDMA Read Type Work Request Completes.

The rules in Section 8.2.2 enable subsequent Work Requests to be executed before the RDMA Read Type Work Request Completes. If however, a delay in processing occurs due to waiting for a prior RDMA Read Type Work Completion, this will effectively prevent subsequent Work Requests from being executed until the delay is over (i.e. stall Send Queue processing). If the Consumer wants to avoid this type of delay in Send Queue processing, it can issue up to as many RDMA Read Work Requests as supported by the value of ORD for that QP, and when each one Completes, then add an additional RDMA Read Type Work Request.

The Consumer should manage the number of RDMA Read Request Messages outstanding, either by correctly setting the QP's ORD value to be less than or equal to the Associated QP's IRD value, or by limiting the number of RDMA Read Type Work Requests the Consumer posts on the Send Queue at any one time to be less than or equal to the Associated QP's IRD value. If this is not done correctly, the Local Peer may attempt to send more RDMA Read Request Messages than the Remote Peer can accept, which will result in an error from the Remote Peer that Terminates the RDMAP Stream (See Section 6.6.2.4 - Remote Termination).

The RDMA Read Resources (IRD and ORD) MUST be initialized at QP creation (Create QP). The RDMA Read Resources MAY be changed while the QP is in Idle state and when the QP is in the RTS state. If the Consumer changes the resources while the QP is in the RTS state, the Consumer should ensure that no RDMA Read Operations are outstanding for the affected direction (outbound for ORD, inbound for IRD). If the Consumer modifies the RDMA Read Resources when RDMA Read Operations are outstanding, the QP state MAY be indeterminate and the RI MUST NOT adversely affect any other QPs supported by the RI. Changing RDMA Read Resources when RDMA Read Operations are not outstanding is easily done if IRD and ORD are set before any RDMA Read Work Requests are posted by either Peer. If RDMA Read Work Requests have already been posted, it is up to the Consumer to ensure that they have all Completed before changing IRD or ORD or the QP may be in an indeterminate state.

The following semantics are required of the RI:

* All RNICs MUST allow the Consumer to reduce the ORD in the IDLE and RTS states.

* It is OPTIONAL for an RI to allow the Consumer to increase IRD or ORD after the QP has been created.

* It is OPTIONAL for an RI to accept reductions of IRD from the Consumer after the QP has been created.

* The RNIC MUST support a total number of inbound RDMA Read Request Messages and outbound RDMA Read Request Messages, so that each is at least equal to the total number of QPs supported by the RNIC. The RNIC thus MUST be able to support at least IRD=1 and ORD=1 for each QP.

* RNICs that implement shared "per RNIC" RDMA Read Resources for IRD and ORD, MUST have enough so that all of the QPs can be assigned a value of one for IRD and one for ORD. It is up to the resource manager to allocate these resources fairly, so that applications that need RDMA Read Resources can be assured of their availability.

Note that the maximum amount of resources returned by Query RNIC may be adversely affected by consumption of unrelated resources, so that not all of the reported number may actually be available simultaneously.

If the Consumer attempts to set either IRD or ORD to one or greater, and there are not enough resources to allow this, the Create QP or Modify QP Verb MUST fail with an Immediate Error.

Note that when using "per RNIC" resources, the Create or Modify QP IRD and ORD values are also limited by the "per QP" resources.

6.5.1 Example IRD/ORD Negotiation

The example in Figure 13 shows one possible negotiation for a single direction (if the ULP uses RDMA Read Operations in both directions on the RDMA Stream, it must also do the same thing in reverse). Note that the last step may be omitted if the ULP is not interested in reducing the resources used at the left side of the connection when the right side supports less.

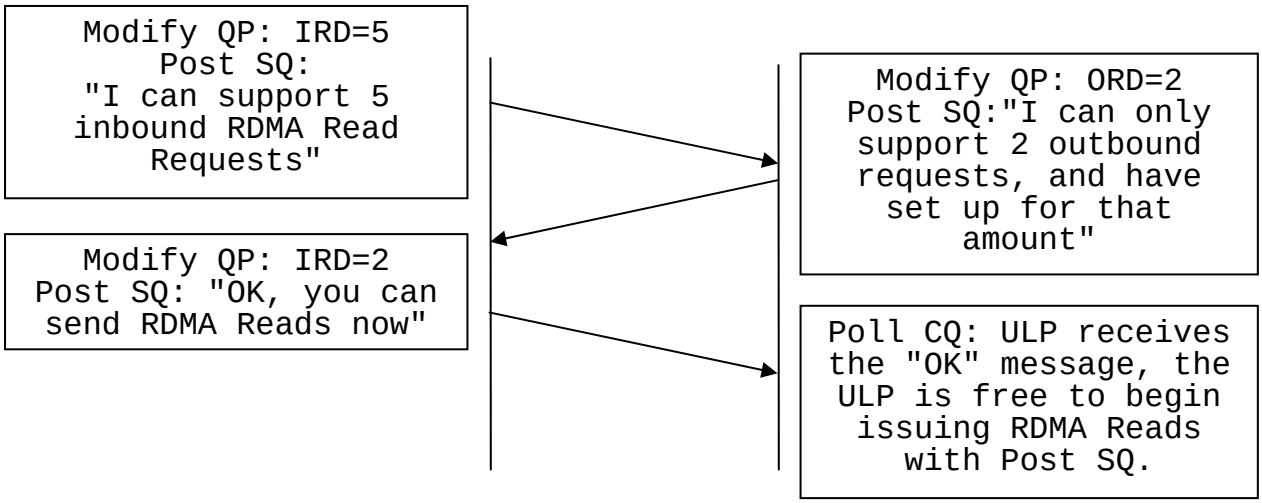


Figure 13 - An example RDMA Read Resource negotiation

6.6 Connection Management

6.6.1 Connection Initialization

RDMA Stream initialization can occur as the transport connection is created or sometime thereafter. In the latter case, the connection may require a ULP supplied end-to-end handshake before iWARP is initialized. Either the active or passive side of the connection may initiate turning on iWARP.

In either case, the ULP must know, before iWARP mode is to begin, which model of operation is to be used by the ULP.

An RI MUST support RDMA Stream initialization sometime after the transport connection is established and some streaming mode data has been sent.

An RI MAY support RDMA Stream startup along with the transport connection, with no streaming mode data sent. This option is more completely described in Section 13.1 - Connection Initialization at LLP Startup.

Once iWARP initialization is complete, the RI MUST allow only iWARP messages to be sent across the LLP connection until the RDMA Stream is torn down.

Section 6.6.1.1 and 6.6.1.2 provide informative examples of methods for the ULP to transition to RDMA mode. Other implementations are possible.

6.6.1.1 Active Connection Initialization after LLP Startup

For this discussion, the Active side goes to iWARP mode first. In the figures below, the thin lines represent TCP Streaming mode and the thick lines represent iWARP mode.

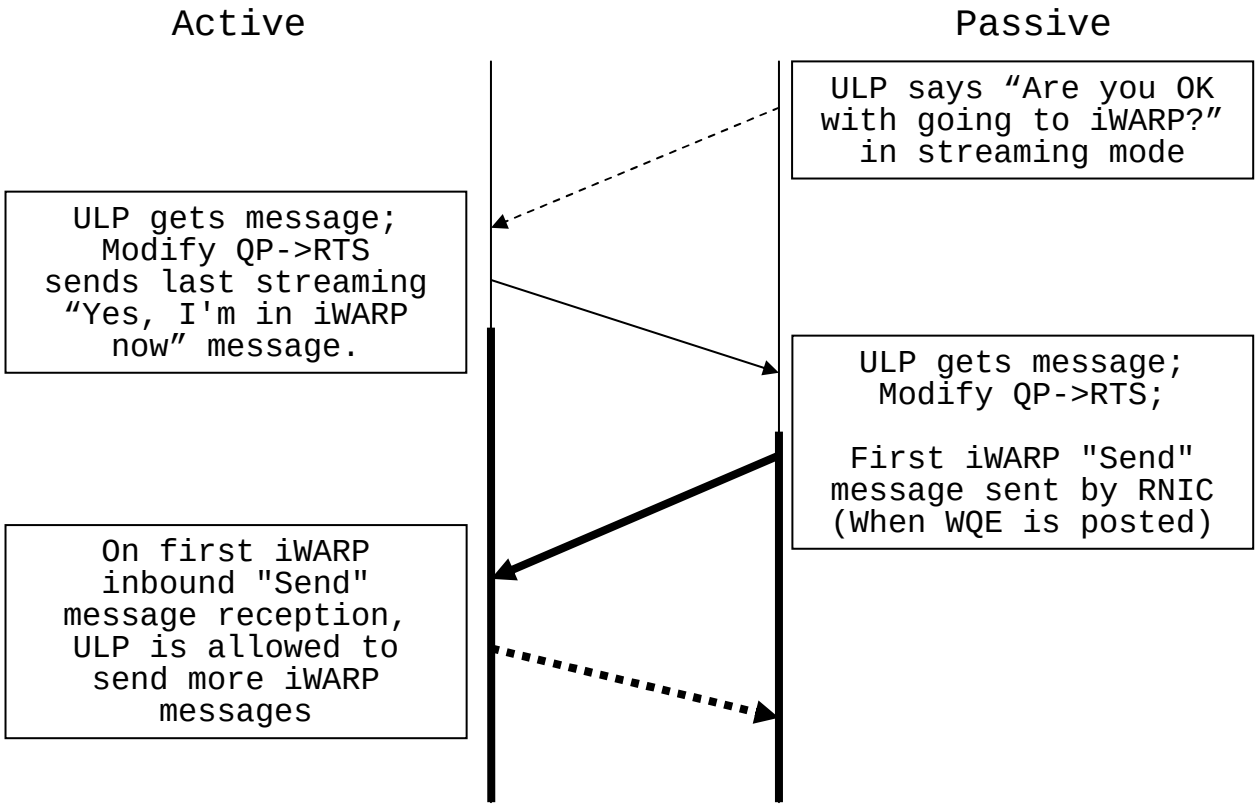


Figure 14 - Connection Initialization after LLP Startup

Below is the sequence for an active side iWARP startup. Note that the dotted line arrows above indicate messages that may not be needed for some implementations.

1. The ULP establishes the LLP Connection and LLP Stream.

2. The active side ULP ensures that the passive side is able to enter iWARP mode via some negotiation or other mechanism, which is outside the scope of this specification.

3. The active side Consumer creates a QP, setting up the CQ, PD etc., and registers memory for buffers. Note that in some instances, this may have been done at some previous time during the initialization process.

4. The active side Consumer posts receive buffers via PostRQ that are appropriate for the expected traffic. A first message may arrive quickly after the transition to RTS.

5. The active side Consumer moves the QP to the RTS state. The Consumer includes the LLP Stream Handle in the Modify QP Verb, and a single message buffer which contains the last streaming mode message to be sent to the Remote Peer. The RI uses the presence of this message buffer to recognize the Active startup sequence. For information on implementing this state transition, see Section 6.2.1.2 - Idle to RTS.

6. When the active side Consumer receives the first RDMA/DDP Message from the passive side (e.g. a Send type message), the active side Consumer is free to post additional Work Requests to the Send Queue. The active side Consumer should not have posted any SQ WRs while the QP was in the Idle state, or while the QP is in the RTS state. The active side consumer should not post any SQ WRs until the first RDMA/DDP Message is received. If the Consumer posts SQ WRs during either of these times, the Remote Peer is likely to improperly synchronize to the LLP Stream and to Terminate the LLP Stream. One way that the Consumer can determine that the message arrives is to have the initial message sent from the Associated QP have the Solicited Event bit set, thus generating an event at the Local Peer.

7. If the local Consumer intends to perform RDMA Read Operations, the local Consumer obtains, by some ULP defined message, the number of Incoming RDMA Read Request Messages that the Remote Peer can have outstanding (IRD). If the Remote Peer's IRD is smaller than the local Peer's ORD, the local Consumer should also perform a Modify QP Verb with the Remote Peer's IRD placed into the local ORD prior to posting the first RDMA Read Type WR. The local Consumer may also transmit, in some ULP defined message, the number of Outbound RDMA Read Request Messages that the Local Peer can have outstanding (ORD).

8. If the local ULP intends the QP to be a target of RDMA Read Operations, the local Consumer provides, in some ULP defined mechanism, the number of Inbound RDMA Read Request Messages that the Local Peer can have outstanding (IRD). The Consumer may also receive, by some ULP defined mechanism, the Number of Outbound RDMA Read Request Messages that the Remote Peer can have outstanding (ORD). If the Remote Peer's ORD is smaller than the Local Peer's IRD and the Local RNIC supports IRD reduction, the

local Consumer could perform a Modify QP Verb with the Remote Peer's ORD placed into the local IRD prior to posting the first RDMA Read Type WR.

6.6.1.2 Passive Connection Initialization after LLP Startup

Below is the sequence for a passive side iWARP startup:

1. The passive side ULP establishes the LLP Connection and LLP Stream.

2. The passive side ULP informs the active side that it is able to enter iWARP mode via some negotiation.

3. The passive side ULP waits for the Active side to send a last streaming mode message to indicate that it should enter RDMA mode and that the remote node is in RDMA mode. When that message arrives, and if it indicates that iWARP mode is desired, the passive side Consumer continues with the items below.

4. The passive side Consumer creates a QP, setting up the CQ, PD etc. Note that this may have been done previously.

5. The passive side Consumer posts receive buffers appropriate for the expected traffic to the RQ.

6. The passive side Consumer posts at least one Send type Work Request that is used by the active side to complete the negotiation. The WR may contain any data that the ULP needs to communicate.

   Note: the passive side Consumer may delay the posting of buffers and Work Requests until after the transition to RTS, described below.

7. The passive side Consumer moves the QP to RTS state, specifying the LLP Stream Handle. The passive side Consumer does not include a last streaming mode message buffer in the Modify QP Verb; if it does, the Remote Peer is likely to improperly synchronize to the RDMA Stream and be forced to terminate the LLP Stream.

8. The passive side Consumer may now begin posting additional Work Requests.

9. If the local Consumer intends to perform RDMA Read Operations, the local Consumer obtains, in some ULP defined message, the number of incoming RDMA Read Request Messages that the Remote Peer can have outstanding (IRD). If the Remote Peer's IRD is smaller than the local Peer's ORD, the local Consumer should

also perform a Modify QP Verb with the Remote Peer's IRD placed into the local ORD prior to posting the first RDMA Read Type WR. The local Consumer may also transmit, in some ULP defined message, the number of outgoing RDMA Read Request Messages that the Local Peer can have outstanding (ORD).

10. If the local Consumer intends the QP to be a target of RDMA Read Operations, the Consumer provides, in some ULP defined message, the number of incoming RDMA Read Request Messages that the Local Peer can have outstanding (IRD). The Consumer may also receive, in some ULP defined message, the number of outgoing RDMA Read Request Messages that the Remote Peer can have outstanding (ORD). If the Remote Peer's ORD is smaller than the Local Peer's IRD, the local Consumer may also perform a Modify QP Verb with the Remote Peer's ORD value placed into the local IRD prior to posting the first RDMA Read Type WR, if the RI supports IRD reduction.

6.6.2 Connection Teardown

Five types of iWARP and LLP connection teardown mechanisms are supported:

* A normal close is an LLP Close that finishes with no errors (see Section 6.2.5 - Closing State, for a list of possible errors). This is used when the Consumers on both sides of the connection have sent their last message and wish to close the LLP Stream (see Section 6.6.2.1 - Normal Close).

* A ULP initiated Termination is used when the ULP desires to perform an LLP Close with an error message to the Associated QP (see Section 6.6.2.2 - ULP Initiated Termination).

* A ULP initiated Abortive Teardown is used when the ULP wishes to perform an LLP Reset with no error message to the Associated QP (see Section 6.6.2.3 - ULP Initiated Abortive Teardown).

* Remote Termination occurs when the RI receives a Terminate Message from the Associated QP, and the LLP Close process has begun (see Section 6.6.2.4 - Remote Termination).

* Local Termination, Local Abortive Teardown and Remote Abortive Teardown occur when the RI or LLP Stream detects an error and a Terminate Message is sent prior to an LLP Close or an LLP Reset is initiated (see Section 6.6.2.5 - Local Termination, Local Abortive Teardown and Remote Abortive Teardown).

Sections 6.6.2.1 through 6.6.2.5 provide informative examples of methods for the ULP to terminate an RDMA Stream. Other implementations are possible.

6.6.2.1 Normal Close

A normal close is provided as a mechanism for the ULP to cease activity, flush any receive buffers that have been posted to the RQ, and disassociate the LLP Stream from the QP. It requires that no errors occur during the close process. If an error occurs, it is now an abnormal close, which would cause the QP to transition to the Error state.

The Consumer initiates a normal close, either locally or remotely, when both sides of a LLP Stream agree to the close.

When the Consumer desires a normal close, the following items must be done:

1. The Consumer waits for all outstanding Work Requests on the Send Queues on both sides of the LLP Stream to be Completed. Note: the Completion on the remote WQ can be inferred by the arrival of a SEND message from the ULP that indicates that it intends to do no more work.

2. One of the Consumers moves the QP state to Closing with the Modify QP Verb, resulting in the following actions:

   - If any WQEs are present on the Send Queue, or if any RDMA Read Operations are incomplete on the IRRQ, an error will result (for more information, see Section 6.2.5 - Closing State).

   - The RI stops QP processing and flushes all incomplete WQEs on the Receive Queue by Completing them with the Flushed Completion Status.

   - The RI performs an LLP Close. If this QP was using the last LLP Stream on the LLP Connection, the RI closes the LLP Connection.

   - When the LLP Close actions are complete, the RI automatically moves the QP to the Idle state and an Affiliated Asynchronous Event: "LLP Close Complete" is created.

3. The Consumer may re-use the QP for a new LLP Stream or it may destroy the QP (see Section 6.1.3 - Modifying Queue Pair Attributes and Section 6.1.4 - Destroying a Queue Pair).

The normal close may also be initiated remotely (e.g. for TCP a FIN segment is received). If the Send Queue is empty and the IRRQ is empty, the RI moves the QP state to the Closing state and an

Asynchronous Event: "LLP Close Complete" will be generated. If this is the last LLP Stream, the LLP Connection will be closed.

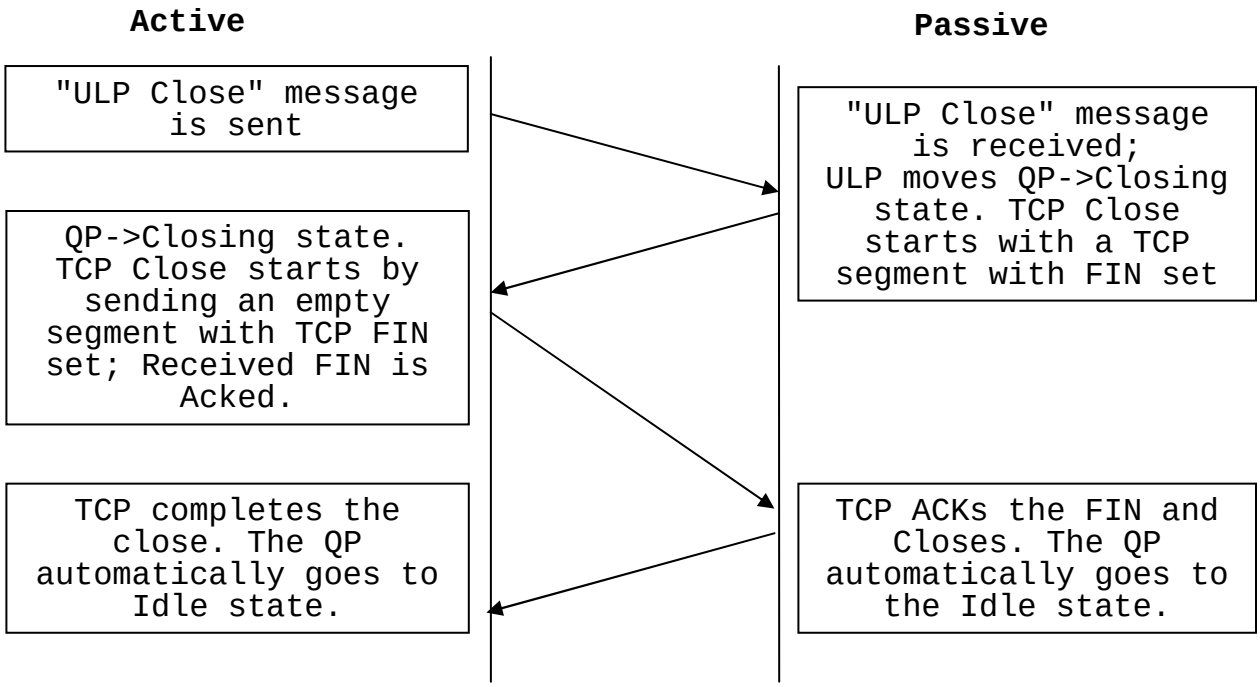


Figure 15 - Normal Close on TCP

6.6.2.2 ULP Initiated Termination

A ULP initiated Termination is usually used when the Consumer (such as the OS) detects an error. The ULP needs to perform an LLP Close, but would like to let the Remote Peer know that an error occurred. Note that an ULP initiated termination may entail loss of data.

When the ULP desires a ULP initiated Termination, the following items must be done:

1. The Consumer modifies the QP to the Terminate state.

    - Before returning from the Modify QP -> Terminate, the RI stops QP processing, formats a Terminate Message containing the termination code: "Local Catastrophic Error" and sends it to the Remote Peer.

    - The RI performs an LLP Close. If the LLP cannot deliver the Terminate Message, an LLP Reset is performed, and the RI generates an Asynchronous Error Event: "Bad Close".

2. After returning from the Modify QP -> Terminate, the Consumer waits for the QP to automatically be moved to the Error state. This is signaled by an Asynchronous Error Event: "Error State Entered".

3. Once in the Error state, the RI flushes all incomplete WQEs on both the Send and Receive Queues by completing them with the Flushed Completion Status. The Consumer would presumably reap all of the Work Completions to ensure all resources are cleaned up. Once the Consumer believes all Work Completions have been reaped, it should attempt to transition the QP to the Idle state by performing a Modify QP. If the transition is successful, the Consumer knows it can either re-use the QP for another LLP Stream or call Destroy QP (see Section 6.1.3 - Modifying Queue Pair Attributes and Section 6.1.4 - Destroying a Queue Pair). If the Modify QP returns with an error (presumably because Work Requests are still being flushed), the Consumer must try at a later time to transition to the Idle state. The Consumer might arm a timeout. If the Consumer is unable to transition to the Idle state after some amount of time, it should destroy the QP (presumably because the QP can not recover from an internal error).

6.6.2.3 ULP Initiated Abortive Teardown

A ULP initiated Abortive Teardown is usually used when the Consumer (such as the OS) detects an error, and the ULP needs to tear down the entire LLP Stream immediately (i.e. perform an LLP Reset). Note that a ULP initiated abortive teardown may entail loss of data.

When the ULP desires an Abnormal ULP initiated Abortive Teardown, the following items must be done:

1. The Consumer modifies the QP to the Error state.

   - The RI stops QP processing and performs an LLP Reset.

2. Once in the Error state, the RI flushes all incomplete WQEs on both the Send and Receive Queues by completing them with the Flushed Completion Status. The Consumer would presumably reap all of the Work Completions to ensure all resources are cleaned up. Once the Consumer believes all Work Completions have been reaped, it should attempt to transition the QP to the Idle state by performing a Modify QP. If the transition is successful, the Consumer knows it can either re-use the QP for another LLP Stream or it can invoke Destroy QP (see Section 6.1.3 - Modifying Queue Pair Attributes and Section 6.1.4 - Destroying a Queue Pair). If the Modify QP returns with an error (presumably because Work Requests are still being flushed), the Consumer must try at a later time to transition to the Idle state. The

Consumer might arm a timeout. If the Consumer is unable to transition to the Idle state after some amount of time, it should destroy the QP (presumably because the QP can not recover from an internal error).

6.6.2.4 Remote Termination

Remote Termination occurs when the Associated QP sends a Terminate Message to the Local Peer. Note that remote termination may entail loss of data.

When the Remote Peer sends a Terminate Message, and it is locally received, the following sequence occurs:

1. The RI stops QP processing.

2. The RI moves the QP automatically to the Terminate state. The RI then generates an Asynchronous Error Event: "Terminate Message Received".

3. The RI performs an LLP Close, or if an LLP final timeout occurs, an LLP Reset.

4. The RI moves the QP to the Error state.

5. Once in the Error state, the RI flushes all incomplete WQEs on both the Send and Receive Queues by completing them with the Flushed Completion Status. The Consumer would presumably reap all of the Work Completions to ensure all resources are cleaned up. Once the Consumer believes all Work Completions have been reaped, it should attempt to transition the QP to the Idle state by performing a Modify QP. If the transition is successful, the Consumer knows it can either re-use the QP for another LLP Stream or it can invoke Destroy QP (see Section 6.1.3 - Modifying Queue Pair Attributes and Section 6.1.4 - Destroying a Queue Pair). If the Modify QP returns with an error (presumably because Work Requests are still being flushed), the Consumer must try at a later time to transition to the Idle state. The Consumer might arm a timeout. If the Consumer is unable to transition to the Idle state after some amount of time, it should destroy the QP (presumably because the QP can not recover from an internal error).

6.6.2.5 Local Termination, Local Abortive Teardown and Remote Abortive Teardown

iWARP defines an abortive teardown mechanism which is invoked if a catastrophic iWARP error is encountered locally. iWARP attempts to send a Terminate Message, but depending upon the condition of the LLP, it is possible a Terminate Message can not be sent or can not

be successfully delivered to the Associated QP. If an LLP Stream error occurs, it is possible for the LLP Stream or LLP Connection to be torn down before a) iWARP is aware of the error, b) before iWARP is able to send the Terminate Message, or c) after iWARP has posted the Terminate Message to the LLP, but it is still in the LLP send queue. Thus the Consumer at the Remote Peer may or may not be able to retrieve a valid Terminate reason for some forms of abortive teardown. The Consumer at the Remote Peer can retrieve the Terminate Message, if available, using the Query QP when the QP has transitioned to the Error state. The Consumer at the Local Peer should always be able to retrieve the Terminate Message that was sent (if the QP transitioned through the Terminate state), regardless of whether it was successfully delivered to the Remote Peer.

Note that an abortive teardown may entail loss of data. The RI will complete all outstanding (incomplete) iWARP messages in error. In general, when an abortive teardown occurs it is impossible to tell for sure what iWARP messages were successfully placed and delivered at the Remote Peer. Thus even completed messages on the Send Queue should be treated as incomplete unless a ULP Acknowledge has been received. Note that Completed RDMA Read Type Work Requests act as a ULP Acknowledgement, in that any prior RDMA Write Messages, Send Type Messages, RDMA Read Operations and the RDMA Read Request Message itself are required to have arrived at the Remote Peer before the RDMA Read Response Message can be generated at the Remote Peer to Complete the RDMA Read Type Work Request.

When iWARP detects a local error the following items are done:

1. If the LLP Stream is still functional, the RI moves the QP to the Terminate state. If the error was not reported in a CQE, the RI generates an Asynchronous Error Event, with an appropriate error code (see 8.3.3 - Asynchronous Errors). Then the RI stops QP processing.

   If the LLP Stream is not functional, the RI performs an LLP Reset and moves the QP to the Error state. If the error was not reported in a CQE, the RI generates an Asynchronous Error Event, with an appropriate error code (see 8.3.3 - Asynchronous Errors). The RI skips steps 2 and 3 below.

2. The RI formats a Terminate Message with an appropriate termination error code and sends it to the Remote Peer.

3. The RI performs an LLP Close. If the LLP could not successfully perform the LLP Close (e.g. for TCP, transitioning through the normal closing states incurred a final timeout), an LLP Reset occurs. Once either the LLP Close or LLP Reset is finished, the RI transitions the QP to the Error state.

4. Once in the Error state, the RI flushes all incomplete WQEs on both the Send and Receive Queues by completing them with the Flushed Completion Status. The Consumer would presumably reap all of the Work Completions to ensure all resources are cleaned up. Once the Consumer believes all Work Completions have been reaped, it should attempt to transition the QP to the Idle state by performing a Modify QP. If the transition is successful, the Consumer knows it can either re-use the QP for another LLP Stream or it can invoke Destroy QP (see Section 6.1.3 - Modifying Queue Pair Attributes and Section 6.1.4 - Destroying a Queue Pair). If the Modify QP returns with an error (presumably because Work Requests are still being flushed), the Consumer must try at a later time to transition to the Idle state. The Consumer might arm a timeout. If the Consumer is unable to transition to the Idle state after some amount of time, it should destroy the QP (presumably because the QP can not recover from an internal error).

Figure 16 is an example of how the abortive teardown might occur. Other sequences of events are possible. For example, the TCP FIN could be sent in a separate TCP segment. Another example is the Remote Peer RI might not transition from the Terminate state when the LLP can no longer be used for data transmission (i.e. the TCP FIN ACK segment is sent). Instead it waits for TCP finite state machine to reach the Closed state. If the latter implementation is used, QP resources may not be able to be recycled until after TCP finishes transitioning through the TIME-WAIT state, which takes a considerable amount of time. See Section 10, Security Considerations, for potential security issues with this approach.

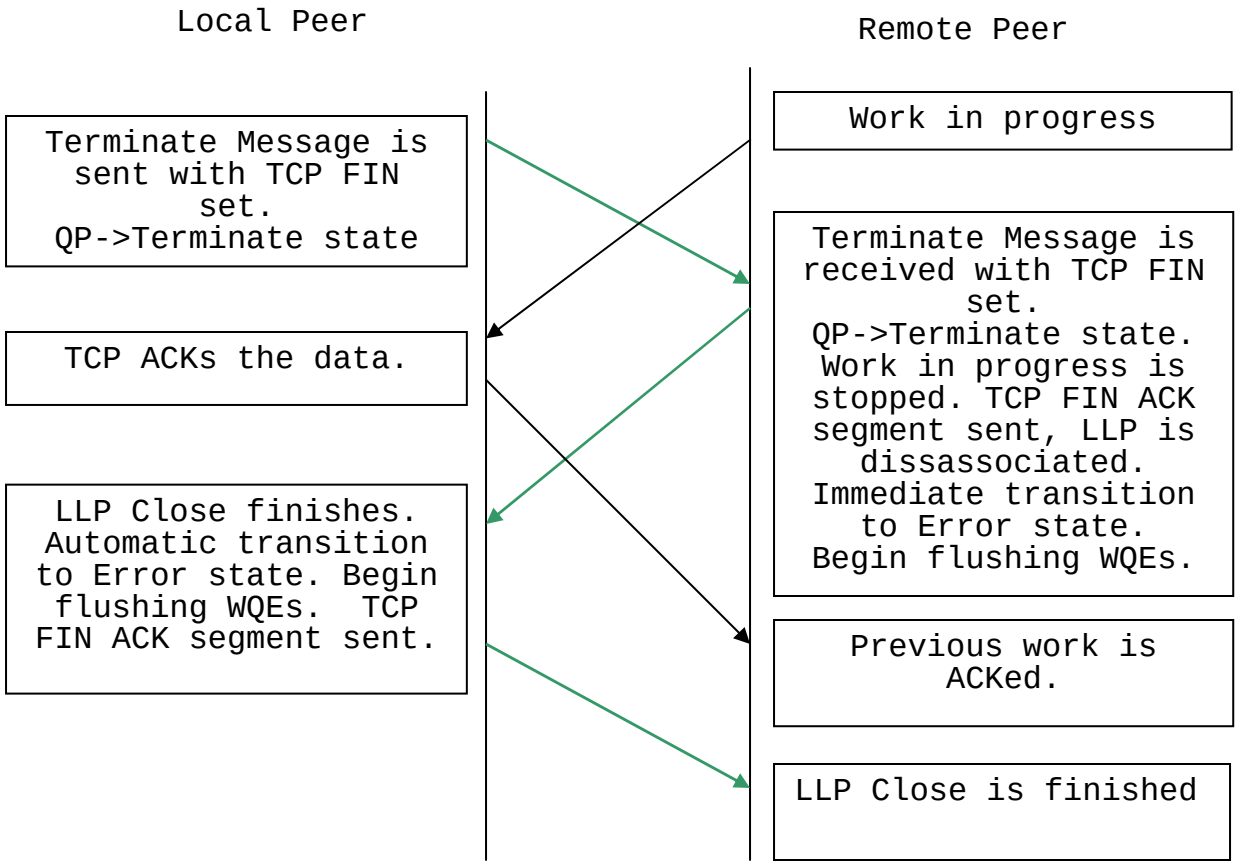


Figure 16 - Abortive Teardown example on TCP

7 Memory Management

7.1 Memory Management Overview

There are two basic methods for enabling memory to be accessed by an RNIC. These are Memory Regions and Memory Windows. Memory Regions are used to assign an STag to a Physical Buffer List, associate it with a starting Tagged Offset and length, and assign it Memory Access Rights. Memory Windows are used to assign an STag to a portion, or window, of a Memory Region.

Fundamental to Memory Management is the definition of an STag (see Section 7.2 - Steering Tag (STag)) and the Tagged Offset (TO) associated with it (see Section 7.3.1.1 - Memory Region Tagged Offset (TO) and Section 7.6.1 - Addressing Registered Memory). Also fundamental is the concept of a Physical Buffer List (PBL), which contains the physical address mappings for the memory used in the Memory Region, as discussed in Section 7.6.2 - Physical Buffer Lists.

An STag can be associated with either a Memory Region or a Memory Window. While both Memory Regions and Memory Windows can be used for data transfer operations, they differ with respect to the Verbs used to manipulate them. These distinctions are covered in great detail in this section.

There are three mechanisms for associating a Memory Region's STag with a Physical Buffer List. A Consumer can allocate an STag with the PBL in one step, as is done with RI-Register Non-Shared Memory Region. A Consumer can also allocate an STag and then use a Fast-Register WR to associate the PBL with the STag. Finally, a Consumer can create a new STag that is associated with an existing Memory Region through the Register Shared Memory Region. For more information on Memory Region creation, see Section 7.3.2 - Memory Region Creation and Registration.

There are two types of Memory Regions. These are Non-Shared MR and Shared MR. A Non-Shared MR has a PBL that is not shared with other MRs. A Shared MR has a PBL that may be shared with other MRs. A Non-Shared MR becomes a Shared MR through the Register Shared Memory Region operation. For more information on Shared Memory Regions, see Section 7.3.2.4 - Register Shared Memory Region. MR (without any qualifiers) is used to refer to both Non-Shared MR and Shared MRs.

Before use, Memory Windows must first be allocated and then Bound to a Memory Region. The allocation is a RI Verbs call, but the Bind operation is a WR. For more information on Memory Windows, see Section 7.10 - Memory Windows.

Memory registration enables access to a Memory Region by a specific RNIC. Binding a Memory Window enables the specific RNIC to access memory represented by that Memory Window. STags are specific to an RNIC and the RI is NOT REQUIRED to grant access to the Memory Region by other local RNICs.

Mechanisms are provided for Re-registering Non-Shared Memory Regions. These are discussed in Sections 7.3.2.3 - RI-Reregister Non-Shared Memory Region. In addition, the Verbs provide mechanisms for Registering Memory Regions which share PBL mappings. These are discussed in Section 7.3.2.4 - Register Shared Memory Region.

Architecturally, only Bind Memory Window and Fast-Register Non-Shared Memory Region are anticipated to be optimized for performance. The rest of the Memory Registration mechanisms are not anticipated to be performance optimized.

All Memory Regions MUST have Access Rights associated with them to indicate if local read, local write, remote read and remote write accesses are allowed. This is discussed in Section 7.4 - Access to Registered Memory. All Memory Windows MUST have Access Rights associated with them to indicate if remote read and remote write accesses are allowed. This is discussed in Section 7.4 - Access to Registered Memory.

Non-Shared Memory Regions and Memory Windows have to be invalidated before they can have their PBL associations changed. This has other benefits as well, such as preventing remote accesses using that STag. This is discussed is Section 7.8 - Invalidating Memory Regions and 7.10.4 - Invalidating or De-allocating Memory Windows.

The RI also provides Verbs for retrieving STag attributes, as discussed in Section 7.7 - Querying Memory Regions and 7.10.3 - Querying Memory Windows. The Verbs also define the destruction and deallocation of Memory Windows and Memory Regions in Section 7.9 - Deallocation of STag associated with a Memory Region and in Section 7.10.4 - Invalidating or De-allocating Memory Windows, respectively.

7.2 Steering Tag (STag)

All local and remote memory accesses through the Verbs require the use of an STag. For local access, the STag, along with a Tagged Offset (TO) is used by the RI, when processing a Work Request’s SGE, to identify a memory location within a specific Memory Region. For remote access, the STag, along with a TO, is used by the RI when handling RDMA operations to identify a memory location within a specific Memory Region or Memory Window.

An STag is a 32-bit identifier that has two sub-fields: a Consumer provided STag Key and an RI provided STag Index. The STag Key is the

8 least significant bits of the STag. The STag Index is the 24 most significant bits of the STag.

The 8 bit STag Key is provided by the Consumer. The Consumer can use the STag Key in any way it desires. For example, it can be used as an incrementing value to help discover application errors by using a different value with each registration. As a general rule, the Consumer provides the STag Key to the RI whenever the consumer causes the transition of an STag to the Valid state, or when the STag is being Invalidated. In the Invalid state, only the STag Index is meaningful.

There is no default value for the STag Key. The RI MUST use the STag Key provided by the Consumer for the following Verbs:

* Register Non-Shared Memory Region,

* Register Shared Memory Region,

* Reregister Non-Shared Memory Region,

* PostSQ Verb Fast-Register Non-Shared Memory Region operation, and

* PostSQ Verb Bind operation,

* PostSQ Invalidate Local STag.

The RI MUST return the value of the STag Index sub-field on an invocation of the following:

* Allocate Non-Shared Memory Region STag,

* Allocate Memory Window,

* Register Non-Shared Memory Region,

* Register Shared Memory Region, and

* Reregister Non-Shared Memory Region.

The RI MUST use the same STag Index sub-field as was passed in by the Consumer, on an invocation of the following:

* Query Memory Region,

* Query Memory Window,

* Register Shared Memory Region,

* Reregister Non-Shared Memory Region,

* PostSQ Fast-Register Non-Shared Memory Region,

* PostSQ Bind Memory Window,

* PostSQ Invalidate Local STag, and

* Deallocate STag.

Implementation Note: To guarantee that the immediately previous STag is no longer valid, the Consumer may change the STag Key field each time the STag is bound. The use of a suitable random number with each binding can provide a valuable interface check and diagnostic tool.

7.2.1 STag of zero

The STag of zero (STag with a value of zero) is a special STag. It has a fixed value for the STag Index and STag Key. The STag Key is composed of all zeros and the STag Index is composed of all zeros. It has no PD associated with it and it cannot be used for Remote Access operations.

The purpose of an STag of zero is to allow Privileged Mode Consumers to be able to reference a Physical Buffer in a WR without first registering the buffer with the RI. This approach has the advantage of reduced overhead. It has the potential disadvantage that the buffer is represented by only a single SGE and therefore must be contiguous. Note that buffers which are not contiguous can be represented by multiple SGEs in this case, but all SGLs have a finite limit of the number of entries allowed by the RI. If the buffer is not physically contiguous, any access to the non-existent memory may result in an access error.

Using an STag of zero as part of a Scatter/Gather Element tells the RNIC that it MUST interpret the TO portion of the SGE as a physical address on the local node. Note the RI MUST never generate an STag Index of zero. The RI MUST NOT allow the Consumer to associate an STag Key with the STag of zero.

The STag of zero has the following semantics, which are different than the semantics of any other STag:

1. The RNIC MUST NOT perform any PD checks on an STag of zero.

2. When accessing an STag of zero on a given QP, the RNIC MUST assure access to the STag of zero is enabled on that QP. If allowing an STag of zero is not enabled, then the operation MUST result in a protection error.

3. The RNIC MUST NOT permit any remote access that references STag of zero and any attempt to do so MUST result in a protection error. The RI MUST grant STag of zero Local Read and Local Write Access Rights.

4. The RNIC MUST NOT allow Memory Windows to be Bound to STag of zero. Any attempt to do so MUST result in an error.

5. The RNIC MUST NOT allow a Local or Remote Invalidation of the STag of zero. Any attempt to do so MUST result in an error. The STag of zero MUST always be in the Valid state.

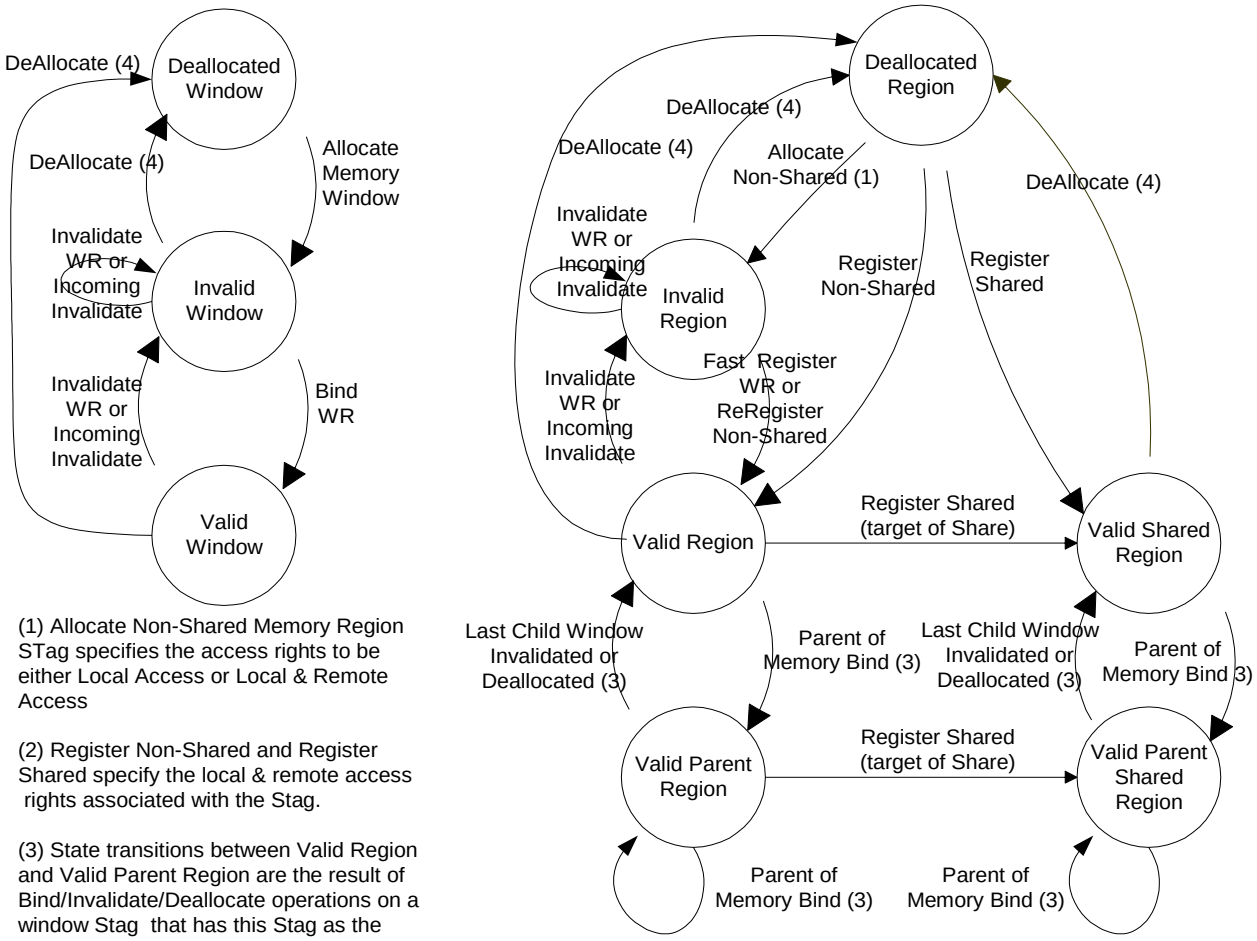
6. The RNIC MUST NOT allow an STag of zero to be an input modifier of an RI-Reregister Non-Shared Memory Region, Register Shared Memory Region, Query Memory Region, Query Memory Window, Bind Memory Window, Deallocate STag, Invalidate STag or Fast-Register and MUST return an Immediate Error if a Consumer attempts to do so.

7. The RI MUST NOT return a value of zero as an STag Index for RI-Register Non-Shared Memory Region, RI-Reregister Non-Shared Memory Region, Register Shared Memory Region, Allocate Non-Shared Memory Region STag and Allocate Memory Window.

7.2.2 Summary of Memory Region STag States

The STag associated with a Non-Shared Memory Region has two states. They are Invalid and Valid. Memory accesses MUST NOT be allowed if the STag is in the Invalid state.

Below in Figure 17 is the Memory Region and Memory Window state diagram. It indicates the state transitions required to change Memory Regions and Memory Windows from the Valid state to and from the Invalid state. In addition, it denotes the effects of the Register Shared Memory Region Verb on a Memory Region.



(1) Allocate Non-Shared Memory Region STag specifies the access rights to be either Local Access or Local & Remote Access

(2) Register Non-Shared and Register Shared specify the local & remote access rights associated with the Stag.

(3) State transitions between Valid Region and Valid Parent Region are the result of Bind/Invalidate/Deallocate operations on a window Stag that has this Stag as the parent region.

(4) Deallocating an Stag revokes all access rights associated with that Stag.

(5) A non-shared region becomes a Shared Region if a Register Shared verb uses the Stag as the target to be shared, i.e. the Stag is an input modifier to Register Shared.

(6) The term "Parent" in this diagram refers to a Memory Region which has one or more Memory Windows bound to it.

Figure 17 - Memory Region and Window State Diagram

For a Non-Shared Memory Region, the following bulleted list indicates the state, if memory access is allowed in that state, and what Verbs are used to enter and exit the specified state.

* Invalid - May not be used to access a memory location.
    - Entered through: Allocate Non-Shared Memory Region STag, PostSQ Invalidate STag, incoming Send with Invalidate STag Message, incoming Send with Solicited Event and Invalidate STag Message, or local RDMA Read with Invalidate Local STag WR.
    - Exited through: RI-Register Non-Shared Memory Region, RI-Reregister Non-Shared Memory Region, Fast-Register Non-Shared Memory Region WR, or Deallocate STag.
* Valid - May be used to access a memory location.
    - Entered through: RI-Register Non-Shared Memory Region, RI-Reregister Non-Shared Memory Region, Fast-Register Non-Shared Memory Region WR.
    - Exited through: PostSQ Invalidate STag, incoming Send with Invalidate STag Message, incoming Send with Solicited Event and Invalidate STag Message, local RDMA Read with Invalidate Local STag WR, or Deallocate STag.

Note: Deallocate STag exits the state logic captured above, as does RI-Reregister Non-Shared Memory Region (if a different STag is returned).

The STag associated with a Shared Memory Region MUST always be in the Valid state. Note that the Register Shared Memory Region Verb does two things – it returns a new Shared Memory Region STag for an existing Memory Region's Physical Buffer List (either Shared or Non-Shared), and if the input STag is for a Non-Shared MR, the Non-Shared MR is permanently converted into a Shared MR (See Section 7.3.2.4 - Register Shared Memory Region). The following bulleted list indicates what Verbs are used to enter and exit the Valid state for a Shared Memory Region.

* Valid - May be used to access a memory location.
    - Entered through: Register Shared Memory Region.
    - Exited through: Deallocate STag.

Note: Deallocate STag of a Non-Shared MR MUST exit the state logic captured above.

7.3 Memory Registration

Memory Registration provides mechanisms that allow Consumers to describe a set of virtually contiguous memory locations or a set of

physically contiguous memory locations to the RI in order to allow the RNIC to access either as a virtually contiguous buffer using the STag and Tagged Offset.

Memory Registration provides the RNIC with a mapping between a STag and Tagged Offset and a Physical Memory Address. It also provides the RNIC with a description of the access control associated with the memory location.

Before using a data buffer with the RI, all Consumers MUST explicitly register with the RI the memory locations associated with the data buffer, except when using an STag of zero. Local or remote attempts to access unregistered memory MUST result in a protection error. Thus every WR simply uses an STag, TO and length to reference a buffer.

Memory Registration MAY fail due to the RNIC's inability to find resources to hold information needed by the RNIC to record the registration. Memory MUST NOT be registered in this case and MUST NOT consume any RI resources if the Registration fails.

7.3.1 Memory Regions

A set of memory locations that have been registered are referred to as a Memory Region (MR).

The RNIC uses two values to identify a memory location within a Memory Region: Steering Tag (STag) and Tagged Offset (TO).

7.3.1.1 Memory Region Tagged Offset (TO)

The base of the TO field is specified by the Consumer when the Memory Region is registered through RI-Register Non-Shared Memory Region, RI-Reregister Non-Shared Memory Region, or Fast-Register Non-Shared Memory Region. Two bases MUST be supported by the RNIC: Virtual Address (VA) based TO and zero based TO. For a VA based TO, the TO of the first memory location associated with the Memory Region equals the VA value passed as an input modifier of the Verb or WR used to register the Memory Region. For a zero based TO, the TO of the first memory location associated with the Memory Region equals zero.

7.3.2 Memory Region Creation and Registration

Before the RNIC can use a Memory Region, the resources associated with a Memory Region must be allocated and the Memory Region must be registered with the RNIC. The RI defines the following mechanisms for providing these functions through the Verbs interface: Allocate Non-Shared Memory Region STag, Register Shared Memory Region, RI-

Register Non-Shared Memory Region, RI-Reregister Non-Shared Memory Region, and Fast-Register Non-Shared Memory Region.

When registering a Memory Region, the Consumer specifies whether Memory Windows may be Bound to the Memory Region or not.

7.3.2.1 Allocate Non-Shared Memory Region STag

This Verb allocates memory registration resources in the RI. When the Verb completes, the STag Index will be allocated as described below and provided as an output modifier.

When allocating an STag:

* the RI MUST verify the Consumer specified maximum Physical Buffer List Size is less than or equal to the size allowed by the RI. The RI MUST return the Physical Buffer List (PBL) size allocated, which MUST be greater than or equal to the size requested. The RI MUST also return the allocated STag Index. If the Consumer specified a maximum PBL Size greater than the size allowed by the RI, the RI MUST return an Immediate Error.

* the RI MUST verify and use the Consumer specified Input Modifier called the Remote Access Flag to indicate if Remote Access is enabled with the STag. If the Remote Access Flag is enabled, the RI MUST be able to allow remote reads or remote writes that reference the STag. Otherwise, the RI MUST NOT allow the STag to be used in remote read or remote write operations.

An STag created through the Allocate Non-Shared Memory Region STag Verb MUST be able to be used in an RI-Reregister or a Fast-Register Non-Shared Memory Region.

When the Allocate Non-Shared Memory Region STag Verb returns control to the Consumer and the Verb has completed successfully, the returned STag is in the Invalid state. The STag MUST be placed in the Valid state before it can be used by a local or remote operation to access a memory location. See Section 7.2.2 - Summary of Memory Region STag States for the requirements on transitioning the STag to the Valid state.

For a description of the Verb which Allocates an STag, see Section 9.2.6.1 - Allocate Non-Shared Memory Region STag.

7.3.2.2 RI-Register Non-Shared Memory Region

When the RI-Register Non-Shared Memory Region Verb returns, it has allocated the appropriate memory registration resources on the RNIC and has registered a Non-Shared Memory Region. When the RI-Register Non-Shared Memory Region Verb is invoked:

* The RI MUST accept and use any STag Key passed in by the Consumer for the Memory Registration.

* The RI MUST use the Physical Buffer List passed in by the Consumer.

* The RI MUST verify and use the Consumer specified modifier which indicates if Remote Access is enabled with the STag. If Remote Access is enabled, the RI MUST allow remote reads or remote writes that reference the STag. Otherwise, the RI MUST NOT allow the STag to be used in remote read or remote write operations.

When the RI-Register Non-Shared Memory Region Verb completes successfully:

* the RI MUST have Registered the Non-Shared Memory Region with the RNIC,

* the RI MUST return the STag Index associated with the Non-Shared Memory Region to the Consumer,

* the RI MUST return the number of Physical Buffer List Entries in the allocated Physical Buffer List, which may be larger than the requested size, and

* the returned STag MUST be in the Valid state.

See Section 9.2.6.2 - Register Non-Shared Memory Region (RI-Register) for a description of the RI-Register Non-Shared Memory Region Verb.

7.3.2.3 RI-Reregister Non-Shared Memory Region

This Verb conceptually performs the functional equivalent of Deallocate STag followed by RI-Register Non-Shared Memory Region. Where possible, resources below the Verb layer are expected to be reused instead of deallocated and reallocated. This Verb may be used to change the Access Rights and/or PD ID of a Region, as well as changing the memory locations that are registered.

When the RI-Reregister Non-Shared Memory Region Verb is invoked:

* The STag MUST be the STag of a Non-Shared Memory Region.

* The STag MUST be in either the Invalid or Valid state.

* The RI MUST accept and use any STag Key passed in by the Consumer for the Memory Reregistration.

* The RI MUST ensure that no Memory Windows are Bound to the STag Index passed in by the Consumer. If any Memory Windows are Bound to it, an Immediate Error is returned.

* The STag passed in by the Consumer MAY have an original PBL size that is smaller than the new PBL size to be associated with that STag. If the PBL passed in by the Consumer is greater than the PBL associated with the STag, the RI MAY return an error indicating it had insufficient resources to complete the request.

If the RI-Reregister Non-Shared Memory Region Verb does not complete successfully:

* If the RI returns an "Invalid RNIC handle", "Invalid STag Index" or "One or more Memory Windows is still Bound to the Region" Immediate Error, the RI MUST make no changes to the current registration (assuming that it even exists).

* If the RI returns any error other than "Invalid RNIC handle", "Invalid STag Index" or "One or more Memory Windows is still Bound to the Region", the RI MUST Deallocate the Memory Region associated with the STag Index used as an Input Modifier and ensure that no new Memory Region is registered.

When the RI-Reregister Non-Shared Memory Region Verb completes successfully:

* the RI MUST have registered the Non-Shared Memory Region with the RNIC;

* the RI MAY return a different STag Index than the one passed in by the Consumer. If a different STag Index is returned, all resources associated with the prior STag MUST have been effectively Deallocated (e.g. transition to the Deallocated state);

* the RI MUST return the number of Physical Buffer List Entries in the allocated Physical Buffer List, which may be larger than the requested size,

* the RI MUST use and set the Remote Access Rights and Remote Access Flag for the STag as indicated with the Input Modifier, and

* the returned STag MUST be in the Valid state. This STag can be used to access a memory location.

The Consumer should note that since the STag Index returned MAY be different than the STag Index provided to the Verb, any attempt to

use the previous STag Index in this case would result in a memory protection error.

The RI-Reregister Non-Shared Memory Region Verb can be used to modify the attributes of a Memory Region created through the RI-Register Non-Shared Memory Region, RI-Reregister Non-Shared Memory Region, or an Allocate Non-Shared Memory Region STag Verb. A Memory Region MUST be allowed to be reregistered an arbitrary number of times provided the PBL length is less than or equal to the original PBL length.

For the error case where a Remote Peer is accessing a Non-Shared Memory Region while it is in the process of being reregistered, implementations MUST present the same semantics as a deallocate or invalidate operation followed by a separate registration operation.

For information on the Verb to Reregister a Memory Region, see Section 9.2.6.5 - Reregister Non-Shared Memory Region (RI-Reregister).

7.3.2.4 Register Shared Memory Region

Shared Memory Regions provide a way for the Consumer to obtain a new STag Index for a Memory Region that has already been registered. This allows optimization of RNIC resources because returning a new STag Index allows the Consumer to assign different Access Rights, change the VA Base, change if the Region is VA Based or Zero Based, assign an STag Key and use a different PD, but use the same Physical Buffer List as a previously registered Memory Region. Thus an optimized implementation is possible where the new STag can use the previous PBL for memory translation but has new STag properties for Access Rights and Protection Domain checks.

When the Shared Memory Region Verb is invoked:

* If the STag Index, passed in by the Consumer, is associated with a Non-Shared Memory Region, the RI MUST verify that the Memory Region STag Index passed in is in the Valid state. Note that Shared Memory Regions are always in the Valid state.

* Any Memory Windows that are currently bound to the MR, associated with the STag Index passed in by the Consumer, MUST be unaffected.

* The RI MUST verify that the STag Key of the existing MR matches the STag Key supplied as an input modifier by the Consumer.

* The RI MUST accept and use any STag Key passed in by the Consumer for the Shared Memory Registration.

* If the STag Index passed in by the Consumer references a VA based TO, the RI MUST verify that the VA passed in by the Consumer produces an FBO that matches the FBO of the PBL that is associated with the STag Index passed in by the Consumer.

When the Shared Memory Region Verb completes successfully:

* the RI MUST have registered the new Shared Memory Region with the RNIC;

* the RI MUST return a different STag Index that is associated with the same or identical PBL as the PBL referenced by the STag Index passed in by the Consumer;

* The RI MUST allow the new Shared Memory Region to have different Access Rights, change the VA Base, change if the Region is VA Based or Zero Based, assign an STag Key and a different PD; and

* if the STag Index passed in by the Consumer is associated with a Non-Shared Memory Region, the RI MUST convert the Non-Shared Memory Region to a Shared Memory Region but MUST NOT change any other attributes of the Memory Region being converted.

The returned STag, which references the new, Shared Memory Region, is in the Valid state. The STag can be used to access a memory location.

7.3.2.5 Fast-Register Non-Shared Memory Region

Fast-Register provides a mechanism for the Consumer to use the PostSQ Verb to invoke an asynchronous memory registration. Fast-Register Non-Shared Memory Region MUST support registration using STags that were created with the Allocate Non-Shared Memory Region STag, RI-Register Non-Shared Memory Region Verb or RI-Reregister Non-Shared Memory Region Verb and have not subsequently been converted to a Shared Memory Region.

When the Fast-Register Non-Shared Memory Region mechanism is invoked:

* The RI MUST accept and use any STag Key passed in by the Consumer for the Fast-Register operation.

* The RI MUST use the STag Index passed in by the Consumer to register a Non-Shared Memory Region with the RNIC.

* The RI MUST verify that the STag Index passed in by the Consumer is in the same PD as the QP. The RI MUST verify that the STag Index passed in by the Consumer is not the STag of zero. The RI MUST verify that the STag Index passed in by the consumer is not

the STag of a Memory Window. If the STag Index is not in the same PD as the QP or the STag is that of a Memory Window or the STag is the STag of zero, the RI MUST return an error.

* The STag MUST be in the Invalid state at the time the Fast-Register Non-Shared Memory Region is processed. See Section 7.2.2 - Summary of Memory Region STag States for more details. If the STag is not in the Invalid state at the time the Fast-Register Non-Shared Memory Region WR is processed, the RI MUST return an error.

* If the Non-Shared Memory Region referenced by the STag does not have a maximum PBL size greater than or equal to the PBL size passed in the Fast-Register Non-Shared Memory Region, the RI MUST return an error.

* The RI MUST prevent an STag with the Remote Access Flag disabled from having its Access Rights changed to include remote Access Rights. The RNIC MUST assure an STag with the Remote Access Flag enabled can have its Access Rights changed to include remote and local, or local only Access Rights. Note that the Remote Access Flag cannot be changed except by the RI-Reregister Non-Shared Memory Region Verb. If Remote Access Rights are requested and the Remote Access Flag is not enabled, the RI MUST return an error.

* The RI MUST verify that Fast-Register access is enabled on the QP that is processing the Fast-Register Non-Shared Memory Region operation. Note that this is intended to prevent a Non-Privileged Mode application from accessing physical memory without Privileged Mode intervention. If Fast-Register is not enabled on the QP, the RI MUST return an error.

The Fast-Register operation MUST take place within the RI at any time between when the Work Request is posted and before execution of the Work Request immediately after the Fast-Register operation.

When the Fast-Register Non-Shared Memory Region operation completes successfully, the associated STag MUST be in the Valid state. The STag can be used to access a memory location.

For a description of the Fast-Register Non-Shared Memory Region mechanism, see Section 9.3.1.1 - PostSQ.

7.4 Access to Registered Memory

The RI MUST support four distinct Memory Region Access Rights: Local Read, Local Write, Remote Read, and Remote Write. The Access Rights of the Memory Region MUST apply to each memory location within the Memory Region. The RI MUST allow changing Access Rights from local

to local and remote only through an RI-Reregister or through a Deallocate followed by an Allocate or RI-Register.

The RI MUST support a Remote Access Flag. It can be supplied as an Input Modifier for the Allocate STag, RI-Register and RI-Reregister Verbs. If the Remote Access Flag is enabled, the RI MUST allow the remote Access Rights to be set on the STag. If the Remote Access Flag is disabled, the RI MUST not allow the remote Access Rights to be set on the STag.

When performing local and remote data transfer operations, the RI MUST validate all 32 bits of the STag used to represent the data transfer.

7.4.1 Local Access to Registered Memory

The RI MUST allow the Consumer to assign one or both of the Local Access Rights to a given Memory Region. If the Consumer does not assign one of the local Access Rights, the RI MUST return an error.

If the RI assigns Local Read Access to a Memory Region, the RNIC is allowed to use the STag and Tagged Offset to read any location within the Memory Region. If the RI assigns Local Write Access to a Memory Region, the RNIC is allowed to use the STag and Tagged Offset to write any location within the Memory Region.

Work Requests may require the Consumer to supply a locally accessible data buffer. Locally accessible data buffers are described by the STag associated with that Memory Region, a Tagged Offset that points to a location within a Memory Region, and the quantity of bytes in the buffer that may be used by the Work Request.

The RI MUST enforce that Scatter Gather Elements used in Send Operation Type and RDMA Write Work Requests posted to the SQ have Local Read Access enabled or a Completion Error will result.

The RI MUST enforce that Scatter Gather Elements used in Receive Work Requests posted to the Receive Queue or Shared-Receive Queue have Local Write Access enabled or a Completion Error will result.

The RI MUST use only Local Access Rights when determining the Access Rights for Scatter/Gather Elements. The RI MUST NOT use Remote Access Rights when determining the Access Rights for Scatter/Gather Elements.

7.4.2 Remote Access to Registered Memory

The Consumer may, in addition to the Local Access Rights, request the RI to assign one or both of the Remote Access Rights to a given

Memory Region. The RI MUST NOT allow the Consumer to assign Remote Write to an MR that has not been assigned Local Write. The RI MUST NOT allow the Consumer to assign Remote Read to an MR that has not been assigned Local Read.

If the Consumer assigns Remote Read Access to a Memory Region, the RNIC is allowed to use the STag and Tagged Offset to read any subset of the Memory Region when processing an incoming RDMA Read Request Message. If the Consumer assigns Remote Write Access to a Memory Region, the RNIC is allowed to use the STag and Tagged Offset to write any subset of the Memory Region when processing an incoming RDMA Write or RDMA Read Response Message. For more information, see [RDMAP].

The RI MUST enforce that Tagged Buffers at the Data Sink targeted by incoming RDMA Write Messages have Remote Write Access enabled or an Asynchronous Error will result at the Data Sink.

The RI MUST enforce that Tagged Buffers whose contents are retrieved by RDMA Read Request Messages have Remote Read Access enabled or an Asynchronous Error will result at the Data Source.

The RI MUST enforce that Tagged Buffers consumed by RDMA Read Response Messages have Remote Write Access enabled or an Asynchronous Error will result at the Data Sink. The access control on the Local Address is not verified until a remote access is attempted through the RDMA Read Response Message.

Remote Access Rights MUST only be used by the RI when determining the Access Rights for incoming Tagged and remote Invalidation operations. The RI MUST NOT allow an STag with only Local Access Rights to be Invalidated by an incoming remote Invalidation operation or a protection error will result.

Figure 18 summarizes local and remote Access Rights and the validity of their combinations that the RI MUST enforce:

Local	Remote	Valid Access Combination
None	None	No
None	Read	No
None	Write	No
None	Read and Write	No
Read	None	Yes
Read	Read	Yes
Read	Write	No
Read	Read and Write	No
Write	None	Yes
Write	Read	No
Write	Write	Yes
Write	Read and Write	No
Read and Write	None	Yes
Read and Write	Read	Yes
Read and Write	Write	Yes
Read and Write	Read and Write	Yes

Figure 18 - Valid Combinations of MR Access Rights

7.4.3 Multiple Registrations of Memory Regions

The same set of memory locations may be registered multiple times, resulting in multiple STags. There are two methods for doing this in the architecture. The first is the Shared Memory Region, which is discussed in Section 7.3.2.4 - Register Shared Memory Region. The second is to simply register a set of memory locations a second time using the same, similar or overlapping Physical Buffer List. Regardless of the method, each resulting STag represents a separate and distinct Memory Region and may be independently associated with any PD and have distinct Access Rights.

The RI MUST support registration of Non-Shared Memory Regions that have partially or completely overlapping Physical Buffer Lists and return a different STag Index for each.

In cases where multiple registrations that use the same memory locations is desired, provision for optimizing the use of RI resources is provided. This Verb is called Register Shared Memory Region and is discussed in Section 7.3.2.4 - Register Shared Memory Region and the Verb is discussed in Section 9.2.6.6 - Register Shared Memory Region.

Given an existing Non-Shared Memory Region, a Shared Memory Region Verb creates a new Shared Memory Region associated with the same Physical Memory Addresses, with the intention that the new Shared Memory Region shares RNIC mapping resources to the extent possible. This also turns the existing Non-Shared Memory Region into a Shared Memory Region. Through repeated calls to the Register Shared Memory Region Verb, an arbitrary number of Shared Memory Regions can potentially share the same RNIC mapping resources, all associated with the same Physical Memory Addresses. The Base TO, VA (if the input STag Index references a VA Based TO), PD ID, and Access Rights specified for the new Shared Memory Region need not be the same as those of the existing Memory Region. For a VA Based TO, the RI MUST verify that the VA passed in by the Consumer produces a FBO that matches the FBO of the PBL that is associated with the STag Index passed in by the Consumer. The lengths are by definition the same.

7.5 Memory Access Control

Only a Privileged Mode Consumer can invoke an RI-Register, RI-Reregister, or Allocate Non-Shared Memory Region STag Verb. In general, the OS is responsible for determining and enforcing access control policy for memory registrations it does on behalf of Non-privileged Consumers. For instance, it is anticipated, but not required, that operating systems will enforce policies similar to the following:

* A Non-Privileged Mode Consumer has control over which of its memory areas can be accessed by local and remote RNIC data transfer operations.

* A Non-Privileged Mode Consumer can enable any local memory area it has access to for access by RNIC data transfer operations.

* A Non-Privileged Mode Consumer cannot enable RNIC read access to memory areas that the Consumer itself doesn’t have read access to.

* A Non-Privileged Mode Consumer cannot enable RNIC write access to memory areas that the Consumer itself doesn't have write access to.

When a Consumer creates QPs or CQs (through the appropriate Verbs), the RI automatically allocates and pins any local memory needed for the associated RI internal control structures. Access by the RNIC to these control structures is implicitly enabled. Access by the Consumer to these control structures is supported only indirectly through Verbs. Any STags used within the RI that are used for the control structures (if they exist) MUST NOT be exposed to the Consumer.

A Consumer controls which Memory Regions and Memory Windows are accessible by each QP through the use of PDs. Prior to creating any QPs, registering any Memory Regions, or allocating any Memory Windows, the Consumer should allocate one or more PDs. When registering Memory Regions or allocating Memory Windows, the Consumer specifies the PD ID to associate to each. For information on the use of PDs, see Section 5.2 - Protection Domains.

7.5.1 Local Access Control

With Send Type, RDMA Write, and Receive Queue WRs, the Consumer explicitly specifies the data buffers to be accessed through the local Scatter Gather Elements (SGEs) that the Consumer posts with the associated Work Requests.

When registering a Memory Region, a Privileged Consumer can generally specify the following local Access Rights for the Region: read only, write only, read and write.

The Consumer can access the Memory Region through the STag. This STag grants the Consumer local Access Rights for the entire Memory Region as bounded by the base TO and byte length and the granularity of the access control is enforced at the byte level.

The following list defines the local Access Rights requirements for SGEs used in local operations:

* Local read access MUST be specified for Gather Elements used in Send Type WRs and RDMA Write WRs,

* Local Write access MUST be specified for Scatter Elements used in Receive WRs, and

* For RDMA Read Type WRs, Local Access Rights are not used to verify the Local Address or Remote Address.

7.5.2 Remote Access Control

When a Consumer wants to allow Remote Peers to access its local memory using RDMA Writes or RDMA Read Operations, the Consumer should explicitly enable remote access and Advertise an appropriate STag to the Remote Peer for it to use when initiating these RDMA Operations targeting the Consumer's (local) memory.

A Consumer can use either of two mechanisms to enable remote access to its memory. The first mechanism consists of using a Memory Region that has remote Access Rights. The second mechanism consists of allocating and binding Memory Windows. Either results in an STag with associated remote Access Rights for the memory referenced by the STag.

Two types of remote access — read and write — are supported. RDMA Write requires Remote Write Access at the Remote Peer. The RDMA Protocol converts an RDMA Read Type WR into an RDMA Read Operation that uses two RDMAP Messages: RDMA Read Request and RDMA Read Response. Remote Read Access MUST be enabled for Memory Regions read by a remote RDMA Read Request Message. Remote Write Access MUST be enabled for Memory Regions written by a remote RDMA Read Response Message. If the Memory Region does not have the appropriate Access Rights, a protection error occurs.

For RDMA Read Operations, during the processing of a RDMA Read Type WR, the RNIC is responsible for generating one RDMA Read Request Message that contains a description of the Local Address and Remote Address. Local Access Rights are not used to verify the Local Address or Remote Address. The Remote Access Rights of the Local Address is not verified until an incoming RDMA Read Response Message is received. The Remote Access Rights of the Remote Address are verified when the Remote Peer processes the RDMA Read Request Message.

In order to set either Remote Access control types in a Fast-Register operation, when the Non-Shared Memory Region STag was created, it MUST have been created with the Remote Access Flag enabled.

7.6 Addressing

The Tagged Offset field is used by local and remote operations to address registered Memory Regions.

7.6.1 Addressing Registered Memory

The RI MUST support two mechanisms for specifying the offset within Memory Regions: VA Based TO and Zero Based TO. At the time the Memory Region is registered, the RI MUST allow the Consumer to

choose between these two mechanisms. A Virtual Address Base Tagged Offset (VA Based TO) is one that has a Tagged Offset base that starts at a non-zero Virtual Address. A Zero Based Tagged Offset (Zero Based TO) is one that has a Tagged Offset base that starts at zero.

7.6.1.1 Addressing with VA based TO

The Virtual Addresses that Consumers manipulate and pass as input modifiers are referred to simply as Virtual Addresses in this specification. The size of the Virtual Addresses used to specify a Memory Region to be registered is implementation dependent. The size of the TO MUST be 64 bits. The TO passed in the SGE defines the VA of the first byte of the SGE.

A Memory Region is specified by a Virtual Address that points to the first byte, which is specified by the First Byte Offset of the Physical Buffer List, and by the length of the set in bytes. The Physical Buffer size that backs the Region depends on the host system hardware and host operating system.

The RI MUST allow a Consumer to specify an arbitrary alignment and length of the virtually contiguous buffer to be registered through a RI-Register Non-Shared Memory Region Verb, RI-Reregister Non-Shared Memory Region Verb, or Fast-Register Non-Shared Memory Region.

The following operations should be performed before registering a VA Based TO Non-Shared Memory Region:

* Translate the set of virtually contiguous memory locations that are associated with the Non-Shared Memory Region into a Physical Buffer List.

* Pin the Physical Buffers in the Physical Buffer List.

While a Memory Region is Valid, every Physical Buffer within the Region must be pinned down in physical memory. This guarantees to the RNIC that the Memory Region is physically resident (not paged out) and that the virtual to physical address translation remains fixed while the Region is registered. The RI is NOT REQUIRED to verify that the Physical Buffers in the Physical Buffer List are pinned.

When the Consumer registers a Non-Shared Memory Region addressed through the VA based TO mechanism, the following input modifiers are passed to the RI (along with additional input modifiers - see Section 9.2.6):

* Virtual Address - The VA Physical Buffer offset portion of the VA defines the offset into the first Physical Buffer of the Non-

Shared Memory Region. The RI checks that the VA modulo Physical Buffer Size equals the FBO.

* Physical Buffer size - Size of all Physical Buffers referenced by the Non-Shared Memory Region.

* First Byte Offset (FBO) - Offset into the first Physical Buffer of the Non-Shared Memory Region

When a RI-Register Non-Shared Memory Region Verb, RI-Reregister Non-Shared Memory Region Verb, Register Shared Memory Region or Fast-Register Non-Shared Memory Region is processed, the RI MUST verify that the Base TO modulo the Physical Buffer Size is equal to the VA modulo the Physical Buffer Size.

7.6.1.2 Addressing with Zero Based TO

A zero based contiguous set of memory locations is specified by the length of the set in bytes. The RI MUST associate a TO that has a value of zero with the First Byte Offset in the Physical Buffer List.

The following operations must be performed before registering a zero Based TO Non-Shared Memory Region:

* Translate the set of virtually contiguous memory locations associated with the Non-Shared Memory Region into a Physical Buffer List.

* Pin the Physical Buffers in the Physical Buffer List.

While a Memory Region is Valid, every Physical Buffer within the Region must be pinned down in physical memory. This guarantees to the RNIC that the Memory Region is physically resident (not paged out) and that the virtual to physical address translation remains fixed while the Region is registered. The RI is NOT REQUIRED to verify that the Physical Buffers in the Physical Buffer List are pinned.

When the Consumer registers a Non-Shared Memory Region addressed through the Zero Based TO mechanism, the following input modifiers are passed to the RI (along with additional input modifiers - see Section 9.2.6):

* First Byte Offset - Offset into the first Physical Buffer of the Non-Shared Memory Region

* Buffer size - Size of all Physical Buffers referenced by the Non-Shared Memory Region.

When a RI-Register Non-Shared Memory Region Verb, RI-Reregister Non-Shared Memory Region Verb, Register Shared Memory Region Verb or Fast-Register Non-Shared Memory Region WR is processed for a Zero base TO MR, the base TO MUST be set to zero.

Note that a Memory Window cannot be bound to a Zero base TO MR.

7.6.2 Physical Buffer Lists

Two Physical Buffer types are defined in this specification: Page and Block. The RI MUST support the Page Physical Buffer type. Support for the Block Physical Buffer type by the RI is OPTIONAL. If the RI supports Block Mode, the RI MUST support the ability to place the RNIC into either Block Mode or Page Mode when the RNIC is opened. The RI MUST support a mechanism for querying the RNIC to determine if the Block Physical Buffer type is supported.

Memory that is part of a Physical Buffer List should remain pinned while the RI has any reference to it. It is not safe for the Consumer to assume that when an STag is deallocated that the Physical Buffer can be unpinned, since another STag may still have a reference to that resource. It is the responsibility of the Consumer to determine if and when the Physical Buffers should be unpinned.

7.6.2.1 Page Lists

A Page List is defined by the following attributes:

* Page size - The size, in bytes, of each page in the list.

* Address List - A list of addresses that point to the physical pages referenced by the Page List. The Address List has the following attributes:

    o All pages in the list have the same size, and that size MUST be a power of two.

    o Page addresses MUST be an integral number of page size. In other words, each address in the Address List modulo page size MUST equal zero.

* First Byte Offset (FBO) - Byte offset to start of Memory Region within the first page.

* Length - Total length in bytes of the Memory Region.

When a Page List is used to register a Non-Shared Memory Region that has a VA based TO, the RI MUST check that the VA modulo the Page Size equals the FBO.

7.6.2.2 Block Lists

A Block List is defined by the following attributes:

* Block size - The size, in bytes, of each block in the list.

* Address List - A list of addresses that point to the physical blocks referenced by the Block List. The Address List has the following attributes:

  - The RI MUST interpret each block referenced in the Address List as having the same size.

  - The RI MUST allow Block Addresses to have an arbitrary byte alignment.

* First Byte Offset (FBO) - Byte offset to start of Memory Region within the first block.

* Length - Total length in bytes of the Memory Region.

When a Block List is used to register a Non-Shared Memory Region that has a VA based TO, the RI MUST check that the VA modulo the Block Size equals the FBO.

7.6.3 Error Checking of Local and Remote Accesses to MRs

When a local or remote operation attempts to access a registered Memory Region, the RI MUST ensure that:

* The Access Rights of the Memory Region allow the type of access being performed by the operation,

* The Access Rights of the QP allow the type of access being performed by the operation,

* For a QP not associated with an S-RQ, the PD ID associated with the Memory Region matches the PD ID associated with the QP that is processing the operation,

* For a QP that is associated with an S-RQ:

  - On an incoming Send Operation Type, the PD ID associated with the Memory Region matches the PD ID associated with the S-RQ that is processing the operation, and

  - On an outbound Send or RDMA Write, or any incoming RDMA Message, the PD ID associated with the Memory Region matches the PD ID associated with the QP that is processing the operation,

* The memory access as specified by the TO & length is within the base and bounds of the Memory Region. The RI MUST enforce this with a byte level granularity.

If the length of the access is zero, the RI MUST NOT perform any of the above checks on the Memory Region.

7.7 Querying Memory Regions

Memory Regions have attributes that can be retrieved through the Query Memory Region Verb. The RI MUST support the complete list of QP attributes as described in Section 9.2.6.3 - Query Memory Region.

7.8 Invalidating Memory Regions

When access to a Non-Shared Memory Region by an RI is no longer required, but the Consumer wants to retain the STag for use in future Fast-Register Non-Shared Memory Region and RI-Reregister Non-Shared Memory Region Verb invocations, the Consumer may directly invalidate access to the Non-Shared Memory Region through an Invalidate Local STag WR or an RDMA Read with Invalidate Local STag WR. Additionally, an STag may be invalidated by a remote Consumer through the use of a Send with Invalidate Message or a Send with Solicited Event and Invalidate Message.

Multiple Memory Regions can represent memory locations that have been registered multiple times. The invalidation of a single STag prevents RNIC access to those memory locations via the STag associated with that Memory Region. Access to the memory locations via STags associated with other Memory Regions other than the STag being Invalidated MUST NOT be affected. Invalidating an STag associated with a Memory Region that partially or completely overlap other Memory Regions MUST NOT cause the RI to affect the registration of those other Memory Regions.

The requirements for unpinning the physical buffers associated with deallocated Memory Regions are covered in Section 7.6.2 - Physical Buffer Lists.

Invalidating an STag associated with a Shared Memory Region MUST result in an Completion Error. Consequently, using an STag associated with a Shared Memory Region under the following conditions will cause a Completion Error at the Data Sink that results in the LLP Stream being torn down after the data transfer operation takes place:

* As the STag specified in an Invalidate Local STag WR.

* As the Data Sink STag for an RDMA Read with Invalidate Local STag WR.

* As the STag to be Invalidated for a Send with Invalidate or Send with SE & Invalidate Message.

When a local Invalidate Local STag WR, a local RDMA Read with Invalidate Local STag WR, an incoming Send with Invalidate, or an incoming Send with Solicited Event and Invalidate completes successfully, the RNIC MUST place the associated STag in the Invalid state. For more information, see Section 8.2.2.1 - Memory Management Operation Ordering.

An Invalidated STag retains associated RI resources, such as the PD, and the Remote Access Flag, and the number of Physical Buffer List entries but the contents of the Address List Entries become indeterminate when the Memory Region is in the Invalid state.

The RI MUST fail Local Work Requests or Remote Operations that attempt to access memory locations in a Non-Shared Memory Region that has had its STag Invalidated with a protection error. The RNIC MUST NOT be able to access any memory locations through an STag that is in the Invalid state.

For Non-Shared Memory Regions created through the RI-Register Non-Shared Memory Region Verb, when an STag is Invalidated, the RNIC MUST retain:

* The Maximum Physical Buffer List (PBL) size and entries used:
    - When the RI-Register Non-Shared Memory Region was invoked, if an RI-Reregister Verb has not been invoked on the Non-Shared Memory Region; or
    - On the last RI-Reregister Non-Shared Memory Region that used the Non-Shared Memory Region.
* The state of the Remote Access Flag.
* The PD associated with the Non-Shared Memory Region.

For Non-Shared Memory Regions created through the Allocate Non-Shared Memory Region STag Verb, when an STag is Invalidated, the RNIC MUST retain:

* The Maximum Physical Buffer List size and entries used:
    - When the STag was created for a Non-Shared Memory Region, if an RI-Reregister Verb has not been invoked on the Non-Shared Memory Region; or
    - On the last RI-Reregister Non-Shared Memory Region that used the Non-Shared Memory Region.

* The state of the Remote Access Flag.

* The PD associated with the Non-Shared Memory Region.

For Memory Regions created through the RI-Reregister Non-Shared Memory Region Verbs, when an STag is Invalidated, the RNIC MUST retain:

* The Maximum Physical Buffer List (PBL) size and entries used:

    o When the RI-Register Non-Shared Memory Region was invoked, if an RI-Reregister Verb has not been invoked on the Non-Shared Memory Region; or

    o On the last RI-Reregister Non-Shared Memory Region that used the Non-Shared Memory Region.

* The PD associated with the Non-Shared Memory Region.

If a Fast-Register is invoked after an RI-Register Memory Region , Allocate Non-Shared Memory Region STag or RI-Reregister Memory Region, the Consumer is guaranteed that the RNIC can register a Non-Shared Memory Region with a PBL size that is equal to or smaller than the original PBL size returned when the Non-Shared Memory Region was created or allocated.

An STag is allowed to already be in the Invalid state, when the RNIC performs the STag Invalidation.

In order to perform an Invalidation Operation on a given QP, either through a Local Invalidation operation or an incoming Send with Invalidate or Send with Solicited Event and Invalidate, the following checks MUST be performed by the RI:

* The STag MUST be Non-Shared and in the Valid or Invalid state.

* The STag MUST NOT be the STag of zero.

* If the STag is that of a Non-Shared Memory Region, the PD ID of the STag MUST equal the PD ID of the QP.

* If the STag is that of a Non-Shared Memory Region, there MUST NOT be any Memory Windows Bound to it.

* The STag Key supplied by the Invalidate Operation must be validated against the STag Key associated with the Memory Region when moving the STag to the Invalid state.

* If the Invalidation Operation is due to an Incoming Send with Invalidate or Send with Solicited Event & Invalidate, the RI

MUST ensure that the QP has either of the remote Access Rights enabled and the STag has either of the remote Access Rights enabled.

If any of the above checks fail, a Protection Error MUST result unless the STag is in the Deallocated state, in which case an Operation Error MUST result. If the operation was initiated by a Local Invalidation, a Completion Error MUST result. If the operation was initiated by an incoming Invalidation operation, a processing error MUST result and the Queue Pair will enter the Terminate state.

For descriptions of the Work Requests that Invalidate STags (Invalidate STag, Send with Invalidate, Send with Solicited Event and Invalidate and RDMA Read with Invalidate Local STag), see Section 9.3.1.1 - PostSQ.

7.9 Deallocation of STag associated with a Memory Region

The Consumer can reverse the allocation or registration process that created the STag by invoking the Deallocate STag Verb. The process of deallocating an STag MUST revoke all RNIC Access Rights associated with that STag.

The RI MUST verify that the STag Index used as an Input Modifier is a valid STag on the specified RNIC.

Multiple Memory Regions can represent memory locations that have been registered multiple times. The deallocation of a single STag prevents RNIC access to those memory locations via the STag associated with that Memory Region. Access to memory locations using STags associated with other Memory Regions MUST NOT be affected. Deallocating an STag associated with a Memory Region that partially or completely overlaps other Memory Regions MUST NOT cause the RI to affect the registration of those other Memory Regions. Deallocating an STag associated with a Shared Memory Region MUST NOT cause the RI to affect the registration of any other Shared Memory Region.

The requirements for unpinning the physical buffers associated with deallocated Memory Regions are covered in Section 7.6.2 - Physical Buffer Lists.

When the Deallocate STag Verb is invoked, any in-process Local or Remote Operations that are actively referencing memory locations by using the STag being deallocated, MUST fail with a protection error. Local or Remote Operations attempting to access memory locations in a Memory Region with a deallocated STag MUST fail with a protection error.

Before the Deallocate Verb returns, the RI MUST free all resources associated with the STag and revoke the right to use the STag in Local or Remote Operations.

When a Deallocate STag is invoked, the RI MUST NOT:

* check the state of the associated STag. That is, an STag associated with a Non-Shared MR can be in either the Valid or Invalid state when the Deallocate STag is invoked.

* check the STag Key portion of the STag. Note that the Deallocate Verb does not have an STag Key Input Modifier.

If any Memory Windows are Bound to the Memory Region and the Consumer invokes the Deallocate STag Verb, the RI MUST return an Immediate Error and MUST NOT deallocate the Memory Region. Memory Windows can reverse the Bind process through deallocation or invalidation.

For a description of the Deallocate Memory Region mechanism, see Section 9.2.6.4 - Deallocate STag.

7.10 Memory Windows

When a Consumer needs more flexible control over remote access to its memory, the Consumer can use Memory Windows. Memory Windows are intended for situations where:

* A Non-Privileged Mode Consumer wants to grant and revoke remote Access Rights to a registered Region in a dynamic fashion with less of a performance penalty than using deallocation/registration or invalidation/re-registration.

* A Consumer wants to grant different remote Access Rights to different Remote Peers and/or grant those rights over different ranges within a registered Region.

To use a Memory Window, the Consumer allocates a Memory Window and then Binds it to a specified TO range of an existing Memory Region that is enabled for use with Memory Windows. The range can include the entire Memory Region or any subset of the Memory Region.

See Section 9.2.6 - Memory Management for a description of the Verbs used to manage Memory Windows.

7.10.1 Allocating Memory Windows

The Allocate Memory Window Verb is used to allocate a Memory Window. When the Verb returns, it must have allocated Memory Window resources on the RNIC, associated the STag with the PD ID supplied

as an Input Modifier by the Consumer, and returned the STag associated with the allocated Memory Window. The RI MUST ensure that the returned STag is in the Invalid state. The RI MUST NOT allow the returned STag to be used with RI-Reregister Non-Shared Memory Region, Register Shared Memory Region, Query Memory Region or Fast-Register Non-Shared Memory Region. For allocating a Memory Window, see Section 9.2.6.7 - Allocate Memory Window.

7.10.2 Binding Memory Windows to Memory Regions

The PostSQ Verb is used to Bind a Memory Window to a previously registered Memory Region. After the WR that Binds the MW is processed, the STag associated with the Memory Window is in the Valid state.

The RI MUST allow a MW to Bind to a Non-Shared Memory Region. The RI MUST allow a MW to Bind to a Shared Memory Region. The RI MUST allow all allocated MWs to be Bound to a single MR. The RI MUST allow all allocated MWs to be Bound to a single QP.

If the STag representing the Memory Region to which the Memory Window will Bind has an STag of zero, the Verb MUST return either an Immediate Error or a Completion Error.

During the processing of PostSQ Bind Memory Window Verb, the RNIC MUST ensure that the PD ID of the Memory Window equals the PD ID of the Memory Region and with the PD ID of the QP that is processing the PostSQ Bind Memory Window Verb. If the three PD IDs are equal, the Memory Window is Bound to the Memory Region and is associated with the QP that processed the PostSQ Bind Memory Window Verb. Otherwise an invalid PD Completion Error is returned to the Consumer. When a Memory Window is Bound to a QP at this point, it is conceptually equivalent to having the PD ID of the Memory Window replaced with the QP ID of the QP. Thus, instead of performing a PD check upon validating the STag for incoming RDMA operations, the QP ID of the Memory Window MUST be equal to the QP ID of the QP where the incoming RDMA operation arrived.

The RI MUST check that the QP has the ability to Bind Memory Windows enabled.

When Binding a Memory Window, the RI MUST ensure that the memory locations being associated with the Memory Window are within the base TO and length of the associated Memory Region. The RI MUST support Memory Windows with a Zero Based TO. The RI MUST support Memory Windows with a VA Based TO. The RI MUST allow Memory Windows to bind to Memory Regions with a VA based TO. If the Memory Window has a VA based TO, the RNIC MUST ensure that the value assigned for the base of the Memory Window be between the MR's base VA, and the MR's Base VA plus the MR's length.

When the Bind MW WR completes successfully:

* The RI MUST have Bound the MW to the Non-Shared Memory Region.

* The RI MUST have Bound the MW to the QP that processed the Bind WR, by associating the QP's QP ID to the MW.

* The RI MUST have set the MW STag's access rights as requested by the Consumer.

* The RI MUST accept and use the STag Key passed in by the Consumer for the Bind operation.

* The RI MUST have set the MW Address Type as requested by the Consumer.

* If the Address Type of the MW was requested as VA Based, the RI MUST have set the Virtual Address as requested by the Consumer.

* The RI MUST have placed the MW STag in the Valid State.

Figure 19 indicates which MR to MW Binding combinations are valid. Note that the figure is based on the Base TO type of the Memory Region and Memory Window. If the Consumer attempts to Bind a MW to a Zero-based TO MR, the RI MUST return an error. The Underlying Memory Region in this case may be either a Non-Shared Memory Region or a Shared Memory Region.

Underlying Memory Region TO base	**Memory Window TO base**	**Valid combination**
Zero based	Zero based	No
Zero based	VA based	No
VA based	Zero based	Yes
VA based	VA based	Yes

Figure 19 - MR to MW Valid Binding Combinations

When a remote access references a Bound Memory Window, the RNIC MUST ensure that the QP ID associated with the Memory Window matches the QP ID associated with the remote access' RDMA Stream. The RNIC MUST also ensure that the memory locations being referenced by the remote access are within the base TO and length of the associated Bound Memory Window. The RI MUST enforce this with a byte level granularity.

When Binding a Memory Window, a Consumer can request any combination of remote Access Rights for the Window. However, if the associated Memory Region does not have local write access enabled and the Consumer requests remote write for the Window, implementations MUST return a Completion Error.

Memory Windows MUST support two distinct remote Access Rights: Remote Read and Remote Write. Bind Memory Window WRs must specify one or both of these rights. Memory Windows with Remote Write Access MUST be bound to Memory Regions that have Local Write Access Enabled. Memory Windows with Remote Read access MUST be bound to Memory Regions that have Local Read Access Enabled.

A Consumer is allowed and commonly expected to enable remote Access Rights when Binding a Window that it may not have enabled when it registered the underlying Region – provided it doesn't violate the above rule regarding local access. For example, a Consumer might register a Region with no remote Access Rights, and later Bind one or more Windows to that Region that would grant remote Access Rights.

Figure 20 summarizes the access right mappings between Memory Regions and Memory Windows and if the Memory Window Access Right requested is allowable or not. The RI MUST validate Memory Windows Access Right requests according to Figure 20 and if the Access Right requested is not allowed, the Bind operation must result in a Completion Error.

Underlying Memory Region's Local Access Rights	**Requested Remote Access Rights for Memory Window**	**Access Right Requested allowed:**
Local Read	Remote Write	No
Local Read	Remote Read	Yes
Local Read	Remote Read and Write	No
Local Write	Remote Write	Yes
Local Write	Remote Read	No
Local Write	Remote Read and Write	No
Local Read and Write	Remote Write	Yes
Local Read and Write	Remote Read	Yes

Underlying Memory Region's Local Access Rights	Requested Remote Access Rights for Memory Window	Access Right Requested allowed:
Local Read and Write	Remote Read and Write	Yes
None	Any	No
Any	None	No

Figure 20 - Valid Combinations of MW & MR Access Rights

Allocating or de-allocating a Memory Window requires a Privileged mode transition for a Non-Privileged Consumer, and thus incurs the associated software overhead. Binding a Memory Window is performed with a Work Request posted to a Send Queue, and thus incurs far less software overhead.

An STag used in a PostSQ Bind Memory Window Verb MUST be in the Invalid state.

Each time a Memory Window is Bound, the Consumer passes the STag Key portion of the STag to the RI. The RI MUST use the STag Key provided by the Consumer. Additionally, the RI MUST NOT change the STag Index portion of the STag passed in by the Consumer. Note that the Bind Memory Window WR has unique ordering rules which are detailed in Section 8.2.2.1 - Memory Management Operation Ordering. Once the Bind operation has completed processing, RNIC implementations MUST guarantee that no additional accesses on this Memory Window can be performed with any STag Key other than the one used in the last Bind operation.

If the RNIC detects an error with the Bind operation, it MUST put the QP into the Error state.

Multiple Windows can be Bound to the same Memory Region, each with arbitrary remote Access Rights, and their associated areas can be overlapping or disjoint.

For a description of the error conditions checked during MW Bind and MW access, see Section 7.10.6 - Error Checking during Memory Window Operations.

For a description of the Bind Memory Window operation, see Section 9.3.1.1 - PostSQ.

7.10.3 Querying Memory Windows

Memory Windows have attributes that can be retrieved through the Query Memory Window Verb. The RI MUST support the complete list of QP attributes as described in Section 9.2.6.8 - Query Memory Window.

7.10.4 Invalidating or De-allocating Memory Windows

When access to a Memory Window by the RI is no longer required, but the Consumer wants to retain the STag for use in future PostSQ Bind Memory Window Verb invocations, the Consumer may directly invalidate access to the Memory Window through either an Invalidate Local STag WR or an RDMA Read with Invalidate Local STag WR. Additionally, an STag associated with a Memory Window may be invalidated by a remote Consumer through the use of a Send with Invalidate Message or a Send with Solicited Event and Invalidate Message. For more information on these Verbs, see Section 7.8 - Invalidating Memory Regions.

Memory Windows are Deallocated in a fashion similar to Memory Regions: with the Deallocate STag Verb. For more information, see Section 7.9 - Deallocation of STag associated with a Memory Region.

When processing an Invalidate operation on an MW STag:

* and the MW is in the Valid state, the RI MUST check and enforce that the QP ID associated with the MW is equal to the QP ID of the QP processing the Invalidate Local STag WR. If the QP IDs match, the RNIC MUST place the specified local STag in the Invalid state. If the QP IDs do not match, the RI MUST return an error.

* and the MW is in the Invalid state, the RI MUST check and enforce that the PD ID associated with the MW is equal to the PD ID associated with the QP processing the Invalidate Local STag WR. If the PD IDs do not match, the RI MUST return an error.

When a local Invalidate Local STag WR, local RDMA Read with Invalidate Local STag WR, an incoming Send with Invalidate Message, or an incoming Send with Solicited Event and Invalidate Message completes successfully, the RNIC MUST:

* transition the associated STag to the Invalid state,

* change the association of the newly invalidated STag from the QP to the PD of the QP that processed the STag Invalidation,

* retain the Memory Window resources associated with the STag,

* remove the association of the Memory Window with the underlying Memory Region.

An invalidated STag which was either Invalidated as described above, or in the Invalid state because it was created through the Allocate Memory Window Verb but never used, can be used as the MW in a PostSQ Bind Memory Window WR.

Once an STag associated with a MW is successfully Invalidated, the RI MUST associate the STag with the PD associated with the QP processing the Invalidate Local STag WR.

For information on Invalidating Memory Windows through the Invalidate Local STag or RDMA Read with Invalidate Local STag WR, see Section 9.3.1.1 - PostSQ. For information on Invalidating Memory Windows through Send with Invalidate or Send with Solicited Event & Invalidate WR, see Section 9.3.1.1 - PostSQ. For a description of the Verb to deallocate a Memory Window, see Section 9.2.6.4 - Deallocate STag.

7.10.4.1 Invalidating or De-allocating Active Windows

Under normal operation, it is improper for a Consumer to deallocate or Invalidate the STag of the Memory Window while it is being used in an incoming, remote operation. However, this can occur if the Remote Consumer misbehaves, or it can occur under error recovery circumstances.

Any Remote Operations that are in-process and actively using a Memory Window when its STag is Invalidated MUST fail with a protection error. Once the Completion of the Invalidate operation has been determined by the Consumer, the RI MUST guarantee that no additional accesses can be performed under the previous binding.

Any Remote Operations that are in-process and actively using a Memory Window when it is deallocated MUST fail with a protection error. Once the de-allocation Verb completes, RNIC implementations MUST guarantee that no additional accesses can be performed through that Memory Window.

An STag is allowed to already be in the Invalid state, when the RNIC performs the STag Invalidation.

7.10.5 Summary of Memory Window STag States

An STag associated with a Memory Window has two states:

* Invalid - May not be used to access a memory location.
    - Entered through: Allocate Memory Window, PostSQ Invalidate STag WR, incoming Send with Invalidate STag Message, incoming Send with Solicited Event and Invalidate STag Message, or local RDMA Read with Invalidate Local STag WR.

- Exited through: PostSQ Bind Memory Window WR or Deallocate STag.

* Valid - May be used to access a memory location.

  - Entered through: PostSQ Bind Memory Window WR.

  - Exited through: PostSQ Invalidate STag MW, incoming Send with Invalidate STag Message, incoming Send with Solicited Event and Invalidate STag Message, local RDMA Read with Invalidate Local STag WR, or Deallocate STag.

Note: Deallocate STag exits the state logic captured above.

7.10.6 Error Checking during Memory Window Operations

7.10.6.1 Error Checking at Window Bind Time

The RI MUST check for the following error conditions during the Memory Window Bind operation and, if any error is detected the RI MUST return a Completion Error.

* The RNIC MUST check and enforce that the MW STag is an MW STag and is in the Invalid state.

* The RNIC MUST check and enforce that the QP has Memory Window Binding enabled.

* The RNIC MUST check and enforce that the STag of the MR is an MR STag and is in the Valid state and is not the STag of zero.

* The RNIC MUST check and enforce that the Memory Window, Memory Region, and QP belong to the same PD.

* The RNIC MUST check and assure that the Memory Region has Window binding enabled.

* The RNIC MUST check and enforce that the Memory Window Access Rights are compatible with the Access Rights of the underlying Memory Region. (See Figure 19).

* The RNIC MUST check and enforce that the Memory Region is not a Zero based TO MR.

* The RNIC MUST check and enforce that the Memory Window base TO and bounds is within the base TO and bounds of the underlying Memory Region. The RI MUST enforce this with a byte level granularity.

7.10.6.2 Error Checking at Window Access Time

The following conditions MUST be checked for each incoming RDMAP Tagged Message targeting an STag that is associated with a Memory Window:

* The RNIC MUST check and enforce that the MW STag is in the Valid state.

* The RNIC MUST check and enforce that the QP ID associated with the Memory Window is equal to the QP ID associated with the incoming remote operation that is accessing the Memory Window.

* The RNIC MUST check and enforce the incoming memory access as represented by the TO and length is within the TO base and bounds of the Memory Window. The RI MUST enforce this with a byte level granularity.

* The RNIC MUST check and enforce the Access Rights associated with the Memory Window.

* The RNIC MUST NOT check or enforce the Access Rights associated with the Memory Region to which the Memory Window is Bound.

* The RI MUST check that the appropriate MW and QP Remote Access Rights are enabled for the incoming RDMA Message. For example, if the incoming RDMA Message is an RDMA Write targeting a MW, the RI must check that the MW and the QP have Remote Write Access Rights enabled.

If any of the above checks fail, the RI MUST not allow the memory access to take place and a protection error MUST be generated.

If the length of the access is zero, the RI MUST NOT perform any of the above checks on the Memory Window.

Note that the QP attributes must be verified as well. For more information, see Section 8.1.2.2.

7.10.6.3 Error Checking at Window Invalidate Time

The following conditions MUST be checked on a PostSQ Invalidate Local STag WR, RDMA Read with Invalidate Local STag WR, incoming Send with Invalidate Message, or incoming Send with Solicited Event and Invalidate Message that accesses a Memory Window:

* If the Memory Window is in the Valid state, the RNIC MUST check and enforce that the QP ID associated with the Memory Window is equal to the QP ID associated with the QP processing the Invalidate Local STag WR.

* If the Memory Window is in the Invalid state, the RNIC MUST check and enforce that the PD ID associated with the Memory Window is equal to the PD ID associated with the QP processing the Invalidate Local STag WR.

If any of the above checks fail, the RI MUST NOT allow the invalidation to take place and the operation MUST result in an error.

8 Work Requests and the WR Processing Model

8.1 Work Requests

A Work Request is the fundamental unit of work used by the Consumer to indicate to the RNIC that there is data to transfer and control operations to process on a specific QP. The following sections describe the creation of Work Requests, types of Work Requests and Work Request Contents.

8.1.1 Creating Work Requests

Work Requests MUST be the only mechanism available to Consumers to submit work to the Work Queues. The Work Requests Verbs MUST be used only to pass operations from the Consumer to the RI. Specifically, these Verbs are PostSQ (Section 9.3.1.1) and PostRQ (Section 9.3.1.2).

Work Requests can only be posted to the SQ or RQ of a specific QP, or, if the QP is associated with an S-RQ, to the S-RQ associated with the QP.

Work Requests are created by the Consumer above the RI and submitted through the Verbs to the RI for processing. The format of Work Requests within the RI is not defined. Its structure is opaque to the Consumer and is not part of this specification. WRs are only valid during the Posting process. WRs are then represented by WQEs until Completed.

The RNIC MUST support the submission of multiple WRs to the RI as a list of individual Work Requests. The intention of this requirement is to allow for optimizations in the RNIC such that the RI can inform the RNIC of WQEs in the most efficient manner for that individual RNIC.

8.1.2 Work Request Types

There are three basic Work Request types. These are those dealing with Send/Receive, RDMA, and Memory.

8.1.2.1 Send/Receive

The Send/Receive model supports the Untagged Buffer Model in the RDMAP/DDP specifications. The Send/Receive model uses a one-to-one correspondence between outgoing Sends Operation Type WRs and incoming Receive Queue WRs. Successful Send Type Work Requests MUST result in the consumption of a Receive Queue Work Request at the Associated QP. Receive Queue Work Requests should be posted to the RQ before the incoming Send Message Type arrives. If a WQE is not available on the RQ to describe the Untagged Buffer for the incoming

Send Message Type, then the LLP Stream MAY be terminated. If the LLP Stream is not terminated, the reader should see Section 13.2 - Graceful Receive Overflow Handling for one implementation option.

The RI MUST allow Send Work Requests to only be posted to a Send Queue. This includes all Send Operation Types, which are: Send, Send with Solicited Event, Send with Invalidate and Send with Solicited Event & Invalidate. The RI MUST allow only Receive Work Requests to be posted to a Receive Queue or Shared Receive Queue.

A Receive Queue Scatter/Gather List Work Request MUST contain at least enough buffer space to place the incoming Send Message Type. If it does not, a Completion Error MUST be returned. The length of the buffer represented by the Scatter/Gather List of a Receive Queue Work Request MAY be greater than the length of the incoming data. The length of incoming data MUST be returned by the RI as part of the Work Completion. In the case of any Completion Error, the value of the length in the Work Completion MUST be considered indeterminate.

Since segmentation and reassembly is provided by DDP, Send Operation Types and corresponding Receives can be larger than the EMSS (See [RDMAP][DDP]). The maximum data transfer length supported by the architecture is 2^32-1 octets of data. Note that for any given message, the length of the buffers represented by the WRs posted to the RQ MAY have a total length that is smaller than the maximum data transfer length. It is up to the Consumer to negotiate the maximum receive buffer size with the Remote Peer.

The Data Source of Send Operation Types MUST be a local Scatter/Gather List. See Section 8.1.3.2 for a description of Scatter/Gather List.

The Data Sink of Receive operations MUST be a local Scatter/Gather List.

8.1.2.2 RDMA

RDMA Write WRs, RDMA Read WRs, and RDMA Read with Invalidate Local STag WRs MUST NOT result in the consumption of a Receive Queue Work Request at the Remote Peer.

The Data Source of an RDMA Write Work Request MUST be a Scatter/Gather List consisting of local buffers.

The Data Sink used in an RDMA Read Type WR MUST be in the local node's address space as represented by the TO, STag and Length contained in the RDMA Read Type WR. The STag MUST be Bound to either a Memory Region or a Memory Window containing the buffer represented by the TO and length.

The Data Source for an RDMA Read Type WR and the Data Sink for an RDMA Write WR MUST be in the Remote Peer's address space as represented by the TO, STag and Length contained in the Work Request. The STag MUST represent either a Memory Region or a Memory Window containing the buffer represented by the STag, TO and length.

Queue Pairs have RDMA Read enable and RDMA Write enable attributes. Memory Regions and Memory Windows have Remote Read and Remote Write attributes as well. Memory Regions also have Local Read and Local Write attributes. RDMA transfers MUST only take place when the appropriate QP RDMA attribute is enabled and the appropriate STag attribute is enabled where the STag represents either a Memory Region or a Memory Window. If the STag is that of a Memory Window, the attributes of the Memory Region do not apply at memory access time. These attributes are checked at the node where the target memory is located. After the STag Access Rights and QP Access Rights have been verified, the RI MUST verify that the STag Access Rights match the QP Access Rights. If the RI detects an invalid Access Rights combination, the operation MUST result in a protection error. The combinations of QP Access Rights and STag Access Rights which will allow the data transfer to take place are shown in Figure 21.

STag Used as	**QP Attribute**	**STag Attribute(5)**	**Access Allowed?**
RDMA Read Type Data Source	Inbound RDMA Read:	Remote Read Access:	
	Enabled	Enabled	Yes
	Disabled	Either	No
	Either	Disabled	No
RDMA Write or RDMA Read Type Data Sink	Inbound RDMA Write and inbound RDMA Read Response:	Remote Write Access:	
	Enabled	Enabled	Yes
	Disabled	Either	No
	Either	Disabled	No
RDMA Write or Send Type Data Source	Either	Local Read Access:	
		Enabled	Yes
		Disabled	No
Receive Data Sink	Either	Local Write Access:	
		Enabled	Yes
		Disabled	No

Figure 21 - Valid QP & STag Access Right Combinations

The RDMA Read with Invalidate Local STag WR behaves similar to an RDMA Read Work Request which is then immediately followed by a Invalidate Local STag WR on the STag in the Local Address. The slight difference in behavior is in this case the Invalidate will not occur until after the RDMA Read Operation is complete; while with two separate WRs, the Invalidate operation could begin processing before the RDMA Read Type WR Completes. Work Requests subsequent to an RDMA Read with Invalidate Local STag WR may begin

---

Footnote 5: The STag may have additional Access Rights, but only the rights listed effect the allowed access.

processing before the RDMA Read with Invalidate Local STag WR Completes. See Section 8.2.2.1 - Memory Management Operation Ordering for more details.

8.1.2.3 Memory

The following Memory Operations can be posted to the SQ: Bind Memory Window, Fast-Register Non-Shared Memory Region, Invalidate Local STag and RDMA Read with Invalidate Local STag.

8.1.2.3.1 Bind Memory Windows

The Bind Memory Window WR associates a previously allocated MW to a specified Tagged Offset (TO) range within an existing MR, as well as sets the MW's RDMA remote Access Rights.

Bind operations MUST be posted to the SQ as a Work Request. Binds only affect local RNIC mapping resources and MUST NOT cause any segment to be issued to the LLP. No resources at the associated QP are directly affected.

For more information on the Memory Window Bind operation, see Section 7.10.2 - Binding Memory Windows to Memory Regions.

8.1.2.3.2 Fast-Register Non-Shared Memory Region

The Fast-Register Non-Shared Memory Region WR associates an MR STag that is in the Invalid state to a specified Physical Buffer List (For more information on Invalidating STags, see Section 7.8 - Invalidating Memory Regions). For information on the STag types allowed, see Section 7.3.2.5 - Fast-Register Non-Shared Memory Region.

Fast-Register Non-Shared Memory Region operations MUST be posted to the Send Queue. Fast-Register Non-Shared Memory Region operations only affect local RNIC mapping resources and do not cause any data transfer. No resources at the Associated QP are directly affected.

8.1.2.3.3 Invalidate Local STag

The Invalidate Local STag and RDMA Read with Invalidate Local STag WRs use the STag supplied as the target for the invalidation and transition the STag to the Invalid state.

The STag which is the target of an Invalidate Local STag or RDMA Read with Invalidate Local STag WR MUST be associated with a Non-Shared Memory Region (i.e. created by Allocate Non-Shared Memory Region STag, RI-Register Non-Shared Memory Region, RI-Reregister Non-Shared Memory Region and has not transitioned to a Shared Memory Region) or MW (i.e. created by Allocate Memory Window).

For information on Invalidating STags associated with a Non-Shared MR, see Section 7.8 - Invalidating Memory Regions. For information on Invalidating STags associated with MWs, see Section 7.10.4 - Invalidating or De-allocating Memory Windows.

Invalidate Local STag operations MUST be posted to the Send Queue as a Work Request. The Invalidate Local STag operations only affect local RNIC mapping resources and MUST NOT cause any data transfer. No resources at the Associated QP are directly affected.

The initiation of an Invalidate Local STag operation must remain ordered with respect to other Work Requests on the same QP and the operation must take effect before any subsequent WRs can begin processing by the RNIC, as defined in the ordering rules in Section 8.2.2.1 and Section 8.2.2.2.

8.1.3 Work Request Contents

Every Work Request submitted through the Verbs contains all of the information required to perform the requested operation. The exact WR contents are covered in the Section 9.3.1.1 - PostSQ and 9.3.1.2 - PostRQ. The characteristics of two of the Post Send Request Verb modifiers are discussed below.

8.1.3.1 Signaled Completions

Signaled Completions refer to Work Requests that result in a Work Completion. Unsignaled Completions provide a mechanism where Work Requests posted to the Send Queue do not generate a Work Completion in the associated Completion Queue if the operations complete successfully. The RI MUST support PostSQ WRs with Unsignaled Completions on every QP.

Every WR posted to the RQ MUST result in a Work Completion. Consequently, all RQ WRs are considered Signaled WRs.

The Consumer can indicate that it does not need a Signaled Completion by setting the Unsignaled Completion indicator in a Work Request posted to the SQ.

When an error is encountered on an Unsignaled or Signaled WR, a CQE will be generated for that WR with the appropriate error code. In addition, the RI MUST Complete all subsequent WRs with a Flushed Error Completion Status regardless of their signaling type. The Consumer is safe in assuming that all WRs prior to the one resulting in an error were completed successfully.

An Unsignaled WR is defined as completed successfully when all of the following rules are met:

* A Work Completion is retrieved from the CQ associated with the SQ where the unsignaled Work Request was posted,

* that Work Completion corresponds to a subsequent Work Request on the same Send Queue as the unsignaled Work Request, and

* the subsequent Work Request is ordered after the unsignaled Work Request as per the ordering rules. Depending on the Work Request used, this may require using the Local Fence indicator in order to guarantee ordering.

When an unsignaled WQE completes successfully:

* The RI MUST free up any resources associated with the Unsignaled WQE,

* The Consumer MAY consider the WQE as having completed successfully, and

* The Consumer MAY re-use any resources associated with the Unsignaled WQE.

The Consumer should ensure that in the event that a WQE with an Unsignaled Completion indicator results in an error that the CQ will not overflow as stated in Section 5.3.1. This is because the WQE will cause a CQE and every WQE after it will cause a CQE as well since they result in CQEs with the Flushed status.

8.1.3.2 Scatter/Gather List

The RI MUST allow each Scatter/Gather List (SGL) to contain one or more Scatter/Gather Elements (SGE). The SGE references a buffer via an STag, TO, and length. The STag specified in the SGE MUST be Registered with the RI prior to submission, except for the STag of zero. These buffers referenced by the STag MUST be considered to be in the scope of the RI from the time they are submitted to a Work Queue until Completion of the Work Request has been confirmed.

If a Memory Window STag is used in an SGE in a PostRQ or PostSQ Send Operation Type or the Data Source for an RDMA Write WR, the RI MUST Complete the Work Request with a Completion Error.

The sum total of all of the buffer lengths in an SGL MUST NOT exceed the maximum message payload size specified for RDMAP. This is 2^32-1 bytes. If an SGE has a length of zero, the STag MUST NOT be validated by the RI. For PostSQ WRs, the sum of the Length field in all of the SGEs MUST be the total length of that RDMAP operation. This value MUST be able to be zero.

An RI MAY support more than one Scatter/Gather Element per Scatter/Gather List. The exact number of Scatter/Gather Elements per Scatter/Gather List supported by the RNIC MUST be returned via the Query RNIC Verb (Section 9.2.1.2) where there is one value for Send Operation Type WR for Data Source buffers (which also applies to PostRQ buffers) and one value for RDMA Write WR Data Source buffers. The Consumer can specify the maximum number of Scatter/Gather Elements per Scatter/Gather List for each Work Queue as an input modifier to the Create QP (Section 9.2.5.1). The RI MUST return an Immediate Error if the value in Create QP exceeds the value supported by the RNIC.

An RI MUST support at least four Scatter/Gather Elements per Scatter/Gather List when the Scatter/Gather List refers to the Data Source of a Send Operation Type or the Data Sink of a Receive Operation. An RI is NOT REQUIRED to support more than one Scatter/Gather Element per Scatter/Gather List when the Scatter/Gather List refers to the Data Source of an RDMA Write.

8.1.3.2.1 STag of zero Usage

The ability to use the reserved STag of zero MUST NOT be allowed for Non-Privileged Mode accessible QPs. The RI must generate an Affiliated Asynchronous Error if an RDMAP Tagged message is received with an STag of zero. If the STag of zero is used in an outgoing RDMA Read Type WR or as the Data Sink of an RDMA Write WR, the RI MUST return a Completion Error. Thus the Consumer should not Advertise the STag of zero, since an error will result.

8.1.3.3 RDMA Data Source & Data Sink

For RDMA Read Type Work Requests, the RI MUST support the Data Source Local Address as an input modifier to PostSQ. The structure representing this information is known as a Data Source Address. A Data Source Address consists of an STag, Tagged Offset and Length. An RI MUST support exactly one Data Source Address for RDMA Read Type Work Requests.

For RDMA Write Work Requests, the RI MUST support the Data Source Scatter/Gather List as an input modifier to PostSQ.

For RDMA Write and RDMA Read Type Work Requests, the RI MUST support the Data Sink Remote Address as an input modifier to PostSQ. The structure representing this information is known as a Data Sink Address. A Data Sink Address consists of an STag, Tagged Offset and Length. An RI MUST support exactly one Data Sink Address for RDMA Read Type Work Requests and RDMA Write Work Requests.

8.2 Work Request Processing Model

The Work Request processing model describes how requests are submitted, processed by the RNIC, and the results returned to the Consumer.

8.2.1 Submitting Work Request to a Work Queue

Work Requests are submitted to the RNIC through the Verbs. They are represented within the RI as Work Queue Elements. Work Queue Elements are abstract. This means they are not accessible directly by the Consumer of the RNIC Interface.

Work Requests can be submitted to the RNIC as a list of Work Requests. Each Work Request in the Work Request List which is successfully inserted into the Work Queue MUST result in the consumption of one WQE on the Work Queue, and each Work Request MUST be submitted to the Work Queue in the order specified in the Work Request List. When a list of WRs containing more than one WR is posted on an SQ, RQ, or an S-RQ, the first Immediate Error in processing a WR MUST stop processing of the Work Request List and MUST NOT enqueue the subsequent WRs in the list onto the Work Queue. All Work Requests prior to the Work Request in error MUST be inserted into the Work Queue. The RI MUST return to the Consumer the number of successfully posted WRs and the verbs result MUST indicate the Immediate Error associated with the WR that resulted in the first error.

The intent of supporting a WR List is to allow some implementations to reduce the number of Consumer to RI interactions when the Consumer has multiple WRs to post, and to reduce the number of interactions between the RI and RNIC due to alerting the RNIC of additional work to perform.

One of the intentions of the architecture is to allow an implementation to pass Work Requests from a Non-Privileged Mode Consumer directly to the RNIC. Consequently, certain Verbs are designed to be invoked in either Privileged Mode or Non-Privileged Mode while others are designed to be invoked only in Privileged Mode. The Verbs that are intended to be invoked in either Privileged Mode or Non-Privileged Mode are: PostSQ, PostRQ, Poll for Completion and Request Completion Notification.

The RI MUST return control to the Consumer immediately after a WR or WR List has been submitted to the SQ, RQ or S-RQ and the RNIC has been notified that a new WR or WR List is ready to process.

The RI MUST ensure that the space occupied by a Work Request in either the Send or Receive Work Queue is not made available for posting a new Work Request until:

* In the case where the WR was Signaled, the associated Completion has been reaped.

* In the case where the WR is Unsignaled, one of the following is true:

  o The WR has Completed processing successfully, OR

  o The associated Completion has been reaped for the WR if the Unsignaled WR Completed in error, OR

  o A Completion associated with a subsequently posted WR to the same WQ has been reaped.

If space is not available on a Work Queue, then an RI MUST return an Immediate Error.

The Unsignaled WR confirmation rules dictate that the Consumer must post a WR with the Signaled Completion indicator set with a frequency less than or equal to the maximum number of WQEs on the SQ. In other words, if X equals the maximum number of WQEs on the SQ, then the Consumer must post at least one Signaled Completion Work Request every X Work Requests. In addition, the Consumer must retrieve a Work Completion of a Signaled Completion with a frequency less than or equal to the maximum number of WQEs on the SQ. This is done in order to force confirmation that prior Unsignaled WRs are Completed. If the Consumer does not follow these rules, a situation may arise where the Consumer is unable to post WRs to the SQ. A ULP reply based on the data that was in a SQ WR is insufficient for determining if the WR has completed, since hardware resources may be held in use until the WCs are polled from the CQ.

The QP can accept Work Requests only when the QP is in a state that allows Work Requests to be submitted.

For details on the Verbs which submit Work Requests, see Sections 9.3.1.1 - PostSQ and 9.3.1.2 - PostRQ.

8.2.2 Work Request Processing

Processing of Work Requests submitted to a Work Queue is initiated and processed according to the rules in this section.

It is important to understand the difference between Placement and Delivery ordering since RDMAP provides different semantics for the two.

Note that many current protocols, both as used in the Internet and elsewhere, assume that data is both Placed and Delivered in order. This allowed applications to take a variety of shortcuts that

depended on in-order Placement and Delivery. For RDMAP, many of these shortcuts are no longer safe to use, and could cause application failure. To ensure reliable operation, applications need to take the rules described below into account.

The following rules apply to implementations of the RDMAP protocol:

1. Send Type, RDMA Write, and RDMA Read Type Work Requests submitted to a Send Queue MUST be initiated and sent in the order submitted to the Send Queue.

2. Work Requests submitted to a single Send Queue or Receive Queue MUST be Completed by the RI in the same order as the Work Requests were submitted. Note that this does not apply to WRs posted to S-RQs.

3. Ordering guarantees for processing and Completion notifications exist only between Work Requests submitted to the same Work Queue. The RI is NOT REQUIRED to provide ordering guarantees across multiple local SQ to remote RQ pairs.

4. RDMA Messages MAY be Placed in any order while in the scope of the RI. If an application uses overlapping buffers (points different Messages or portions of a single Message at the same buffer), then it is possible that the last incoming write to the Data Sink buffer will not be the last outgoing data sent from the Data Source.

5. For a Send Type Operation, the contents of the Receive Queue Buffer at the Data Sink MAY be indeterminate until the Receive Queue Work Request is Completed at the Data Sink.

6. For an RDMA Write Operation, the contents of the buffer at the Data Sink MUST be considered indeterminate until a subsequent Send Type Message is Completed by consuming a Receive Queue WQE at the Data Sink.

7. For an RDMA Read Operation, the contents of the buffer at the Data Sink MUST be considered indeterminate until the RDMA Read Type Work Request has been Completed.

   Statements 5, 6, and 7 imply no peeking at the data in a buffer to see if all of the data has arrived. It is possible for some data to arrive before logically earlier data does, and peeking may cause unpredictable application failure

8. Except for Unsignaled WRs that complete successfully, the resources associated with a Work Request must be considered to be in the scope of the RI from the time the Work Request is submitted to a Work Queue until the associated Work Completion has

been returned. For Unsignaled WRs that complete successfully, refer to Section 8.1.3.1 for a description of when the resources associated with the Unsignaled WR are freed.

9. If the Consumer or Application modifies the contents of Data Sink Buffers while the buffers are in the scope of the RI, the state of the Data Sink Buffers is indeterminate.

10. If the Consumer or Application modifies the contents of Data Source Buffers while the buffers are in the scope of the RI, the state of the Data Sink buffers is indeterminate.

11. The RI is NOT REQUIRED to guarantee that the Completion of an RDMA Write or Send Type WR at the Local Peer means that the ULP Message has: reached the Remote Peer, reached the Remote Peer ULP Buffer, or been examined by the Remote Peer ULP.

12. Incoming Untagged RDMAP Messages (sent in FIFO and MSN order) MUST use RQ or S-RQ Buffers and Complete through the RQ's CQ, in the same order as the Send Message Type Work Requests are posted to the Associated QP's Send Queue.

13. Upon local Completion of an incoming Untagged RDMAP Message the RI MUST guarantee that any prior Send or RDMA Write Messages from the same Associated QP have also Completed at the Data Sink.

14. If the Consumer overlaps its Data Sink buffers for different operations, subsequent Operations MAY cause the RI to overwrite the data in those buffers before the Consumer receives and processes the Completion.

15. The RI MAY begin processing subsequent Work Requests posted to the Send Queue (except for operations which are affected by a fence - see Section 8.2.2.2), before Completing a prior RDMA Read Type Work Request (including zero-length RDMA Read Type Work Requests). Therefore, when an application does an RDMA Read Type Work Request followed by an RDMA Write or Send Type WR targeting the same buffer, it MAY return the data from the later RDMA Write or Send Type WR in the RDMA Read Operation Data Sink buffer, even though the operations Complete in order on the Send Queue's Completion Queue. If this behavior is not desired, the Local Peer Consumer must set the Read Fence indicator on the later RDMA Write or Send Type Work Request.

16. Before an Inbound RDMA Read Request Message is processed (the specified buffer is read), the RI MUST have delivered all prior incoming RDMAP Messages initiated from the same Remote Peer's Send Queue. Therefore, when an application does an RDMA Write or Send Type Work Request followed by an RDMA Read Type Work

Request targeting the same remote buffer, the RDMA Read Type WR MUST return the data as modified by the prior operations.

17. The RI MAY Complete incoming Send Message Types before the RI has finished generating RDMA Read Response Messages for an incoming RDMA Read Request Message (initiated from the same Remote Peer's Send Queue). Therefore, indeterminate results may occur if an application does an RDMA Read Type Work Request followed by a Send Type Work Request, and uses the Work Completion on the Associated QP's RQ Completion Queue (for the incoming Send Type Message) as an indicator that the inbound RDMA Read Operation processing has finished. If this behavior is not desired, the Local Peer Consumer must set the Read Fence indicator on the later RDMA Write (or Send Type) Work Request.

18. If more RDMA Read Type Work Requests are posted to the Send Queue than are indicated by the ORD QP Attribute, the RI MUST pause the processing of the Send Queue until at least one prior RDMA Read Type WR Completes. If zero outbound RDMA Read Request Messages are supported on the QP, and the Consumer posts an RDMA Read Type Work Request, the RI MUST Complete the Work Request in error.

Access by the RNIC to Memory Regions or Memory Windows are NOT REQUIRED to be cache-coherent. If an RNIC caches some portion of memory buffers during the time that the buffers are being processed by the RNIC, there is no requirement that updates to these buffers by any entity be seen by the RNIC. Also, any updates to these buffers by the RNIC are implementation dependent and may not be immediately seen by the system processor, other IO devices, or other RNICs.

8.2.2.1 Memory Management Operation Ordering

This section defines the ordering constraints imposed on Work Requests. The next section defines additional ordering constraints that can be placed by using the Read Fence or Local Fence indicator.

Because one of the objectives of DDP is to enable placement of incoming out-of-order DDP segments into the buffer provided by the Consumer, ordering semantics can not be guaranteed for certain operation combinations. If the Work Request sends payload to the Remote Peer, just because a Work Request Completes locally does not necessarily mean that the Remote Peer has received the data, or that subsequent DDP Segment payload can not overwrite the current data if targeting the same Remote Peer buffer.

Thus, for example, an RDMA Write Message, containing payload1 immediately followed by an RDMA Write Message containing payload2 to the same Remote Peer buffer location may result in the remote buffer

containing either payload1, payload2, or some combination of payload1 and payload2. Thus a programming model that does multiple RDMA Write WRs into the same Remote Peer buffer location without an end-to-end synchronization mechanism is NOT RECOMMENDED.

1. An Incoming Remote Invalidate (the Invalidate portion of the Send with Invalidate or Send with Solicited Event & Invalidate operation) MUST be performed after the Send Message payload is delivered to the appropriate Receive Queue Entry buffer, and before the Associated RQ WR Completes.

   Note: Send with Invalidate is usually used by Remote Peers to invalidate STags that were enabled for remote access and advertised to the Remote Peer. The expected usage is:

   a. Local Peer Consumer creates a Send WR containing a command to be remotely executed and an STag enabled for Remote access and posts it to the Send Queue.

   b. Remote Consumer gets the Send Message through a Completion of an RQ buffer, and does one or more accesses to the STag's buffers via RDMA Read Type WRs and/or RDMA Write WRs.

   c. Remote Consumer creates a Send with Invalidate or Send with SE and Invalidate WR with the status from the Consumer's operation and the original STag to be invalidated as an input modifier. Note that the Read Fence indicator would most likely be set on the Send with Invalidate or Send with SE and Invalidate WR if the remote buffer to be Invalidated was accessed using an RDMA Read or RDMA Read with Invalidate Local STag WR.

   d. RI at Local Peer gets the Send with Invalidate or Send with SE and Invalidate Message, places the data according to the RQ WQE, Invalidates the STag, and creates a CQE on the Receive Queue's Completion Queue, which also contains the Invalidated STag as part of the CQE.

   e. Local Consumer checks that the Invalidate STag output modifier from the Work Completion is the same as was originally sent (as a check on the remote Consumer). If it was not, and the Consumer wishes to prevent remote access, the Consumer should post an Invalidate Local STag WR for the STag.

2. RDMA Read with Invalidate Local STag
   The Invalidate portion of the RDMA Read with Invalidate Local STag Work Request MUST be performed after the RDMA Read Response Message is delivered to the Data Sink buffers, and before a Work Completion is retrieved for the RDMA Read with Invalidate Local

STag WR. As with RDMA Read, subsequent operations MUST be allowed to begin executing before the Invalidate takes place, unless the subsequent operations have the Read Fence indicator set.

3. Fast-Register
   The RI MUST ensure that the Fast-Register operation takes effect prior to the execution of any subsequent Work Requests.

4. Bind
   The RI MUST ensure that the Bind Memory Window operation takes effect prior to the execution of any subsequent Work Requests.

5. Invalidate Local STag
   The Invalidate Local STag Work Request MUST take effect prior to the execution of any subsequent Work Requests.

The RI MAY perform Fast-Register WRs, Bind WRs and Invalidate Local STag WRs at any time between the posting of the Work Request and the execution of a subsequent Work Request. Consequently, it is up to the Consumer to ensure that the posting of the Invalidate Work Request takes place after the STag is no longer in use.

SQ processing of Memory Management Operations (Fast-Register, Bind and Invalidate Local STag) does not usually require the prior operation to Complete before the current operation begins execution. Thus it is possible to have an Invalidate Local STag operation be applied to an RDMA Write WR Data Source buffer before the RDMA Write Message payload has been completely sent. To ensure that this does not occur, the Local Fence indicator may be set to require that all prior operations Complete first (See Section 8.2.2.2).

Note that performing a Fast-Register on an already registered region, or a Bind on a Window that is already Bound, will result in a Completion Error. As such, it is up to the application to ensure that the STag is in the Invalid state before the Fast-Register or Bind Memory Window Work Request is posted.

The rules for Invalidate and Fast-Register or Bind Memory Window above are based on the following usage model:

a. Allocate an STag (through either Allocate Non-Shared Memory Region STag or Allocate Memory Window).

b. Fast-Register or Bind the STag

c. Use the STag in a manner compatible with its Access Rights.

d. Wait for the Completion of the operations using the STag. This ensures that the STag and its related buffer is no longer in use.

e. Invalidate the STag

f. Loop to (b) as long as the STag is still needed; otherwise, Deallocate the STag.

8.2.2.2 Read Fence and Local Fence Indicators

Two types of fence indicators are defined in Verbs – a Read Fence indicator for RDMA Write or Send Type WRs, and a Local Fence indicator for Invalidate Local STag WRs. The Read Fence ensures that the current WR does not execute until all prior RDMA Read Type WRs Complete. The Local Fence indicator ensures that all prior operations Complete before the Invalidate Local STag WR is executed.

Note that in the Verbs specification, a fence indicates that some set of prior operations have completed before the current operation begins. A different concept is operations that are required to Complete before future operations in the SQ can be executed - specifically Bind, Fast-Register, and Invalidate Local STag WR. By default, these operations do not ensure prior operations have completed before they execute. For Invalidate Local STag, if the Local Fence indicator is set, it can ensure that all prior SQ operations Complete before it executes.

Note that RDMAP does not provide any end-to-end acknowledgement except for an RDMA Read Operation. Thus in general an end-to-end fence is not possible without using an RDMA Read Operation, unless an explicit ULP exchange of messages is done. Some operations are local only operations - specifically PostSQ Invalidate Local STag, Bind Memory Window and PostSQ Fast-Register. For combinations of these operations and the local buffers which they operate on (the Data Source for an RDMA Write and Send Type Operation, or the Data Sink for an RDMA Read Operation), it is possible to ensure that a current operation is not executed until prior operation which operate on the referenced local buffer are Completed.

Figure 22 shows the fencing semantics when one operation is followed by another, and whether that operation will not execute until all prior operations have Completed, some prior operations have completed, or potentially no prior operations have completed. The rows are the first operation, and the columns are the second operation. The fields are defined as follows:

* NA-1 - a fence is not applicable. An Invalidate must precede Bind or Fast-Register. Thus in terms of potential WRs in the SQ,

it is the Invalidate Local STag operation that must be fenced to ensure proper operation.

* NA-2 – A fence is not applicable. This is because RDMAP allows RDMA Write Message payloads and Send Type Message payloads to be Placed out-of-order. Thus a local Completion of prior WRs does not ensure the payload has been Placed at the Remote Peer.

* Not Needed – A fence is not needed, because RDMAP requires that the RDMA Read Request Message at the Data Source (i.e. the Remote Peer) must be executed in order. Note that RDMAP does not ensure that operations which are sent after the RDMA Read Request Message occur after the RDMA Read Type WR Completes. Thus the need for the Read Fence Indicator for RDMA Write and Send Type WRs.

* Yes, Full – If the Local Fence indicator is set on the Invalidate Local STag WR, then the operation and subsequent operations will not be executed until all prior operations Complete. Note that this can effectively cause a pipeline stall in transmission of RDMAP Messages, and should be used judiciously.

* Yes, Partial – If the Read Fence indicator is set on the RDMA Write or Send Type WR, then all prior RDMA Read Type WRs must Complete before the current operation can begin execution.

PostSQ Work Request	Send Type	RDMA Write	RDMA Read	Bind	Fast-Register	Invalidate
Send Type	NA-2	NA-2	Not Needed	NA-1	NA-1	Yes, full
RDMA Write	NA-2	NA-2	Not Needed	NA-1	NA-1	Yes, full
RDMA Read	Yes, Partial	Yes, Partial	Not Needed	NA-1	NA-1	Yes, full
Bind	NA-2	NA-2	Not Needed	NA-1	NA-1	Yes, full
Fast-Register	NA-2	NA-2	Not Needed	NA-1	NA-1	Yes, full
Invalidate	NA-2	NA-2	Not Needed	NA-1	NA-1	Yes, full

Figure 22 - Fencing on Prior Operations

The following paragraphs provide the rules which dictate the above behavior.

Read Fence - set in RDMA Write or Send Type Work Requests to ensure all prior RDMA Read Type WRs have been processed by the RI. The RI MUST provide a Read Fence indicator for Send Type Work Requests and RDMA Write Work Requests. This indicator MUST cause the RI to pause before the execution of the Read Fenced Work Request if all prior RDMA Read Type Work Requests are not complete. Once all prior RDMA Read Type Work Requests are complete the RI MUST resume SQ processing.

Local Fence - set in Invalidate Local STag Work Requests to ensure all prior operations have been processed by the RI. The RI MUST provide a Local Fence indicator for the Invalidate Local STag Work Request. This Indicator MUST cause the RI to wait until all prior Work Requests on the Send Queue Complete. Once all prior WRs on the SQ complete, the RI MUST resume SQ processing.

Note: This indicator may be used by the Consumer when there are insufficient STags available to allow them to remain in use until the Consumer can process the Completions for Work Requests using those STags. For example, the following sequence could be used:

a. Allocate an STag (either Allocate Non-Shared Memory Region or Allocate Memory Window)

b. Fast-Register or Bind the STag

c. Use the STag in a manner compatible with its Access Rights.

d. Invalidate the STag using an Invalidate Local STag Work Request with the with Local Fence indicator set.

e. Loop to (b) as long as the STag is still needed; otherwise, Deallocate the STag by invoking the Deallocate STag Verb.

Using this model, the application can reuse an STag multiple times without having to wait for the prior Work Request to Complete before posting the next Work Request. Using the Local Fence indicator may require the RI to stall before processing the Invalidate Local STag Work Request, reducing the rate of Send Queue processing.

Implementation of an end-to-end fence - using an RDMA Write WR followed by an RDMA Read Type WR.
An end-to-end fence ensures that all outstanding operations have been flushed from the network fabric prior to the next operation executing. [RDMAP] enables an application to use an RDMA Read Operation to ensure that all RDMA Write Operations and Send Type Operations prior to the RDMA Read Operation on the same RDMAP Stream have made it to remote memory and can be read back by any other RDMAP Stream connecting through the same remote RNIC with access to the remote memory. The RDMA Read Operation need not be to any of the data written, and can even be a zero length RDMA Read Operation (which does not even require a valid Data Source STag) to have this effect. This enables the Consumer to implement an end-to-end fence by waiting for a RDMA Read WR Completion to determine that data is up to date at the Remote Peer.

If the requirement, for example, is to ensure, from the Data Source, that one RDMA Write Message has been Placed at the Remote Peer before another RDMA Write Message occurs, the following sequence can be used by the Consumer:

a. Perform one (or more) RDMA Write WR(s).

b. Perform an RDMA Read Type WR (zero length is acceptable)

c. Perform a second RDMA Write WR with the Read Fence indicator enabled on the Work Request.

8.2.3 Completion Processing

A CQE is an internal representation of the Work Completion. The results from a Work Request operation are placed in a Completion

Queue Entry (CQE) on the CQ associated with the Work Queue when the request has completed. A CQE MUST be generated for each WQE that results in a Work Completion.

8.2.4 Returning Completed Work Requests

All Work Completions are abstracted through the Verbs. The only method of retrieving a Work Completion MUST be through the Poll for Completion Verb. The RI MUST enable the Consumer to be able to retrieve WCs resulting from WRs posted to QPs which are in any valid QP state. Note that a destroyed QP is not in a valid QP state. See Section 6.1.4.

A Work Request is confirmed Complete when the associated Work Completion is retrieved from its CQ. The RI MUST NOT return a Work Completion for an Unsignaled Work Request that completed successfully. When the RI returns a single WC through Poll for Completion, it MUST free at least one CQE. Note that more than one CQE may be freed due to Unsignaled Completions. See Section 8.1.3.1, Signaled Completions, for the rules on determining when Unsignaled Work Requests have Completed.

When a Work Request has Completed, any Scatter/Gather Elements or other information associated with the original WR are no longer in the domain of the RI. The RI MUST NOT access any memory locations referenced by the Scatter/Gather Elements, Local Address or Remote Address for a WR that has Completed. The RI MUST provide Work Completions through the Poll for Completion Verb no more than once per Work Request. Note that if Destroy QP is invoked with Work Requests pending, the Work Completion may be lost.

The Work Completion contents are specified in 9.3.2.1 - Poll for Completion.

A Consumer is able to find out if a Work Completion is available by polling or notification.

Work Completions MUST be returned when the Consumer polls the CQ in the following cases:

* On Completion of a Work Request submitted to a Send Queue with a Signaled Completion.

* On Completion of a Work Request submitted to a Send Queue that completed in error.

* On Completion of a Work Request submitted to a Receive Queue.

When the Consumer desires to know if a QP has had all of its WRs retrieved and the Work Queues are empty, but there may be only

Unsignaled Work Requests on the Send Queue, the Consumer can transition the QP to the Error state (See Section 6.2.4) and then to the Idle state. This will guarantee that all WRs have been Completed. In order to ensure that the WQEs have been freed and the entries on the CQ have been made available, the Consumer should free any associated CQEs, if any are consumed. There are three methods for a Consumer to free the CQE consumed within the CQ. They are:

* for the Consumer to poll the CQ (See Section 9.3.2.1 - Poll for Completion (Poll CQ)) until the CQ is empty, or

* the Consumer retrieves a WC for a WR submitted to a Work Queue associated with the same CQ where the former WR was submitted and the new WR was submitted after the previous QP was destroyed, or

* the Consumer polls (See Section 9.3.2.1 - Poll for Completion (Poll CQ) a number of Work Completions equal to the total number of entries that the CQ can hold.

8.2.5 Asynchronous Completion Notification

A Consumer of a CQ may request asynchronous notification of when CQEs have been added to a Completion Queue by invoking the Request Completion Notification Verb. The Verbs architecture assumes a Privileged Mode intermediary will process Asynchronous CQ Events for CQs. The Verbs architecture allows this intermediary to register one or more CQ Event Handlers for Asynchronous CQ Events by invoking the Set Completion Event Handler Verb. It is the responsibility of this intermediary to create the asynchronous completion notification to the Consumer that called the Request Completion Notification Verb.

A Completion Event Handler Identifier delineates each Completion Event Handler. The Set Completion Event Handler is invoked once per supported Completion Event Handler. Note that the maximum number of supported Completion Event Handlers is returned by Query RNIC.

Each Set Completion Event Handler invocation can be used to:

* Return a Completion Event Handler Identifier that is used as an input modifier to Create CQ (to associate a CQ with a Completion Event Handler).

* Clear a Completion Event Handler associated with the Completion Event Handler Identifier.

* Modify the address of the Completion Event Handler for the Completion Event Handler Identifier.

The RI is NOT REQUIRED to disassociate CQs from CQ Event Handlers when those CQ Event Handlers associated with the Completion Event Handler Identifiers are cleared. If a CQ Event Handler is cleared and the Consumer still has CQs associated with that CQ Event Handler (through the CQ Event Handler Identifier), and a Completion occurs which would have invoked the CQ Event Handler, behavior of the RI is indeterminate. The Consumer should keep this in mind before clearing the association to prevent indeterminate behavior, such as possible race conditions.

The Request Completion Notification Verb is set on a per CQ basis. When armed, the RI MUST generate at most one notification until the notification has been rearmed by invoking Request Completion Notification Verb. Once Completion Notifications have been enabled, additional Request Completion Notification calls have no effect. The Completion Event Handler will be called only once when the next CQE is added to the CQ. The RI MUST invoke the Completion Event Handler associated with the CQ Event Handler Identifier which is associated with the CQ where the CQE was added. Once the Completion Event Handler routine has been invoked, the Consumer should call Request Completion Notification again to be notified when a new entry is added to the CQ, since the notification is a "one shot" mechanism.

Existing CQEs on the CQ at the time the notification is enabled do not result in a call to the Completion Event Handler. The Completion Event Handler MUST be called when the next CQE is added to the CQ after the Request Completion Notification has been set.

The RI MUST provide the ability for the Consumer to specify whether the Completion Event Handler is invoked for either:

* the next Solicited Completion Event only, or

* the next Completion Event.

If the local Consumer requests the next solicited Completion in the Request Completion Notification Verb, the RI MUST generate a Completion Event when:

* an incoming Send with Solicited Event or Send with SE and Invalidate successfully causes a Receive Queue's WQE to be consumed, and thus a CQE to be added to a CQ, or

* a Work Completion for a Work Request which Completed in error is added to a CQ.

If the Consumer requested an event for the next completion in the Request Completion Notification Verb, the RI MUST generate a Completion Event when any incoming Send operation type or Signaled Local SQ WR completes.

If multiple calls to Request Completion Notification have been made for the same CQ and at least one of the requests set the type to the next Work Completion, the RI MUST invoke the CQ event handler when the next CQE is added to that CQ. The CQ Event Handler MUST be called only once, even if multiple CQ notification requests were made prior to the Completion Event for the specified CQ.

The RI MUST ensure that the following sequence of events will not result in a Completion Notification being missed. Therefore, the following sequence of calls should be used by the Consumer when using Request Completion Notification in order to ensure that a new CQE is not missed for the specified CQ:

* Call Poll for Completion to dequeue all existing CQ entries

* Call Request Completion Notification.

* Call Poll for Completion to dequeue all of the CQ entries that were added between the time the last Poll for Completion was called and the notification was enabled.

When the Completion Event Handler is invoked, the RI MUST supply the CQ handle of the CQ which generated the Completion notification.

The Consumer is responsible for polling the CQ to retrieve the Work Completion. This function MUST NOT be performed automatically by the RI when the notification occurs.

For details on the Asynchronous Completion Verbs, refer to Section 9.4.1 - Set Completion Event Handler and Section 9.3.2.2 - Request Completion Notification.

8.3 Error Handling

The following section details many of the errors that can occur when using the RNIC, and the responsibilities of the RNIC and the Consumer.

Errors are returned to the Consumer by one of three mechanisms: Immediate Errors, Work Completions, or Asynchronous Error Events. Immediate Errors are returned immediately as an Output Modifier of a Verb. Work Completions are used when the error can be related directly to the Work Request in progress. Asynchronous Error Events are used when the error can only be localized to the QP, CQ or RNIC but are not directly attributable to any single Work Request. Each of these errors is described below.

8.3.1 Immediate Errors

Immediate Errors are those surfaced as Verb results provided to the Consumer via Output Modifiers. The individual Immediate Errors are documented within each Verb in Section 9 - RNIC Verbs. A summary of all of the Immediate Errors are covered in Section 9.5.1 - Immediate Status Codes.

When the RI returns an Immediate Error, the RI MUST NOT affect the RI Resource that is the subject of the verb for which the Immediate Error is being returned, except for RI-Reregister Non-Shared Memory Region (which has slightly different rules). That is, for an Immediate Error returned on any verb that has the:

- RI as the subject, the RI remains unchanged;

- CQ as the subject, the CQ remains unchanged;

- QP as the subject, the QP remains unchanged;

- S-RQ as the subject, the S-RQ remains unchanged;

- STag as the subject, the STag remains unchanged (except certain rules for RI-ReRegister Memory Region);

- PD as the subject, the PD remains unchanged;

- Asynchronous Event handling as the subject, Asynchronous Events must not be lost.

8.3.2 Work Completion Errors

The following errors can be associated with a specific Work Request. The RI MUST return a Completion Error via a Work Completion on the Completion Queue associated with the Send or Receive Queue on which the Work Request was posted for the errors defined in Figure 23. The Work Completion's Completion Status field contains the Error information. In each case, the QP MUST be moved to the Terminate state and a Terminate Message is sent with the indicated Terminate code (see Section 6.6.2.5 - Local Termination, Local Abortive Teardown and Remote Abortive Teardown). On any Work Completion that includes the sending of a Terminate Message, the Terminate Message Buffer MUST be available for examination while the QP is in the Terminate state or Error state using Query QP. The Terminate Message may contain useful diagnostic information, depending on the error. For information on the format of the Terminate Message, see [RDMAP].

Error	Terminate Code	Action
Receive Queue Work Request Errors - These errors are probably due to a local Consumer error.		
Invalid WQE format, Invalid STag in SGE, Base and bounds violation (including length errors), Access Rights violation, Invalid PD ID, Wrap error (TO & Segment Length caused an address to wrap).	0x0000	The RI Terminates the LLP Stream with Local Catastrophic Error and the QP transitions to the Terminate state.
Receive Queue Remote Protection Errors - These errors may be due to a Consumer error at either end.		
Invalidate STag Invalid.	0x0100	The RI Terminates the LLP Stream with the indicated Error and the QP transitions to the Terminate state.
Invalidate STag Access Rights.	0x0102	
Invalidate STag Invalid PD ID. or STag not Bound to QP.	0x0103	
Invalidate MR STag had Bound MW.	0x0109	
Send Queue Work Request Errors - These errors are probably due to a local Consumer error.		
Invalid WQE format, Zero ORD.	0x0000	The RI Terminates the LLP Stream with Local Catastrophic Error and the QP transitions to the Terminate state.
Local SQ Protection Errors - Send Types, RDMA Writes, and RDMA Read Types: Invalid STag, Base and bounds violation (including length errors), Access Rights violation, Invalid PD ID, Wrap error.	0x0000	The RI Terminates the LLP Stream with Local Catastrophic Error and the QP transitions to the Terminate state.

Error	Terminate Code	Action
SQ Fast-Register errors: QP not in Privileged Mode, Invalid Region STag, Invalid Physical Buffer Size, Physical Buffer List too long, STag not in Invalid state, Invalid PD ID, Invalid Access Rights Specified, Invalid Virtual Address, Invalid FBO, Invalid Length	0x0000	The RI Terminates the LLP Stream with Local Catastrophic Error and the QP transitions to the Terminate state.
SQ Bind errors: Invalid Region STag Invalid Window STag Base and bounds violation Access Rights violation STag not in Invalid state MR not in Valid state Invalid PD ID	0x0000	The RI Terminates the LLP Stream with Local Catastrophic Error and the QP transitions to the Terminate state.
SQ Invalidate errors (Footnote 6): Invalid STag Invalid PD ID (or QP ID) Invalidate MR STag had Bound MW	0x0000	The RI Terminates the LLP Stream with Local Catastrophic Error and the QP transitions to the Terminate state.

Figure 23 - Completion Errors with Resulting Terminate Codes

8.3.3 Asynchronous Errors

The Consumer may register an Asynchronous Event Handler to be called when an Asynchronous Event occurs which is not associated with an individual CQE by using the Set Asynchronous Event Handler Verb.

An input modifier to the Set Asynchronous Event Handler Verb is the address of the event handler routine. This is a Consumer routine that is invoked when an Asynchronous Event is generated. When the handler routine is invoked, an indication of the origin of the error, called an Event Record, is provided.

Footnote 6: This includes RDMA Read and Invalidate.

The errors defined in Figure 24 are returned to the Consumer via an Event Record in the Asynchronous Event Handler.

There is only one Asynchronous Event Handler per RNIC. If Set Asynchronous Event Handler Verb is called more than once, the new handler MUST replace the previous handler. The RI MUST turn off Asynchronous Event Notification if the Asynchronous Event Handler's address is zero.

After the Asynchronous Event Handler is registered, all subsequent asynchronous events not associated with a CQE MUST result in a call to the handler. Until an Asynchronous Event Handler is registered, asynchronous events will be lost.

For more information, see Section 9.4.2 - Set Asynchronous Event Handler and Section 9.5.3 - Asynchronous Event Identifiers.

The following table covers the errors that can be associated with a QP, thus the Event Record should include the QP ID when the error is associated with a specific QP. On any Asynchronous Error Event that includes the reception or sending of a Terminate Message, the Terminate Message Buffer is available for examination while the QP is in the Terminate or Error state by retrieving it through Query QP. Note that Terminate Messages generated locally as well as Terminated Messages received from the Associated QP are available through Query QP. The Terminate Message may contain useful diagnostic information, depending on the error. For information on the format of the Terminate Message, see [RDMAP].

Error	Terminate Code	Action
Remotely detected Errors		
"Terminate Message Received" An incoming Terminate Message has arrived.	None	QP -> Terminate state. See 6.6.2.4 Remote Termination

Error	Terminate Code	Action
LLP Errors - Errors on incoming RDMAP Segments or Messages probably due to the Remote Peer or fabric corruption.		
"LLP Connection Lost" - Usually caused by Timeout or Too many Retries at the LLP.	None	QP -> Error state. See 6.6.2.4 Remote Termination
"LLP Connection Reset" - Caused by an incoming Reset at the LLP.	None	QP -> Error state. See 6.6.2.4 Remote Termination
"LLP Integrity Error: Segment size invalid" - The incoming segment is too small to contain a valid RDMAP header, or larger than supported by this implementation.	0x1000	If this cannot be corrected by the LLP (drop and retry etc.), then QP -> Terminate state. The RI Terminates the LLP Stream with the indicated error. See 6.6.2.5.
" LLP Integrity Error: Invalid CRC" - The incoming segment had a bad LLP CRC.	0x0202	
"Bad FPDU" -The incoming segment Received MPA marker and 'Length' fields do not agree on the start of a FPDU	0x0203	

Error	Terminate Code	Action
Remote Operation Errors - Protocol Errors on incoming RDMAP Segments or Messages probably due to the Remote Peer.		
Invalid DDP version	0x1206	QP -> Terminate state. The RI Terminates the LLP Stream with the indicated error. See 6.6.2.5.
Invalid RDMA version	0x0205	
Unexpected Opcode	0x0206	
Invalid DDP Queue Number	0x1201	
Invalid RDMA Read Request - RDMA Read not enabled	0x1201	
No 'L' bit when expected	0x0207	
Remote Protection Errors (not associated with the RQ) - Protection Errors on incoming DDP Segments or RDMAP Messages that are not RDMA Read Request Messages, probably due to the Remote Peer's Consumer.		
Invalid STag	0x1100	QP -> Terminate state. The RI Terminates the LLP Stream with the indicated error. See 6.6.2.5.
Base and bounds violation	0x1101	
Access Rights violation	0x1102	
Invalid PD ID	0x1102	
Wrap error - TO and segment length caused an address wrap past 0xFFFFFFFFFFFFFFFF	0x1103	

Error	Terminate Code	Action
Remote Closing Error - Probably due to Consumer not properly synchronizing the ULP close operation.		
Bad Close - QP in Closing state and: Segment arrives, at least one SQ WQE on the SQ, or RDMA in progress.	None	QP -> Error state.
Bad LLP Close - LLP Close received AND (the Send Queue was NOT empty OR the IRRQ was NOT empty) (Footnote 7)	0x0207	QP -> Terminate state. The RI Terminates LLP Stream with indicated error. See 6.6.2.5.
Remote Protection Errors associated with the Receive Queue - Protection Errors on incoming RDMAP Segments or Messages probably due to the Remote Peer's Consumer.		
Invalid MSN – MSN range not valid	0x1202	QP -> Terminate state. The RI Terminates LLP Stream with indicated error. See 6.6.2.5.
Invalid MSN – gap in MSN	0x1202	QP -> Terminate state. The RI Terminates LLP Stream with indicated error. See 6.6.2.5.
IRRQ Protection Errors – Error processing an incoming RDMA Read Request and generating the outgoing RDMA Read Response.		
Invalid STag	0x0100	QP -> Terminate state. The RI Terminates the LLP Stream with the indicated error. See 6.6.2.5.
Base and bounds violation(includes RDMA Read Request larger than supported by the Data Source STag)	0x0101	
Access Rights violation	0x0102	
Invalid PD ID	0x0103	

Footnote 7: For TCP this would be a 1/2 close and a Terminate Message could be sent. For SCTP, no Terminate Message is sent.

Error	Terminate Code	Action
Wrap error - TO and length caused an address wrap past 0xFFFFFFFFFFFFFFFF	0x0104	
Invalid MSN - too many RDMA Read Request Messages in process	0x1203	
Invalid MSN - gap in MSN (RDMA Messages found missing when LLP claims a Message is delivered.)	0x1203	
Invalid MSN - MSN range is not valid (MSN is unreasonably beyond the end of the queue.)	0x1203	
Local Errors		
CQ/SQ error - An error occurred on the CQ during a SQ completion. CQ Overflow error CQ Operation error	0x0207	QP -> Terminate state. The CQ number itself must be determined by using Query QP. The RI Terminates the LLP Stream with the indicated error. See 6.6.2.5.
CQ/RQ error - An error occurred on the CQ during a RQ completion. CQ Overflow error CQ Operation error	0x0207	
S-RQ error on a QP – An error occurred while attempting to pull a WQE from the S-RQ associated with the QP.	0x0207	QP-> Terminate state. The S-RQ can be determined by using Query QP. The RI Terminates the LLP Stream with the indicated error. See 6.6.2.5.
Local QP Catastrophic Error - An error related to the QP occurred while processing (probably a problem with the RNIC).	0x0207	The RI will attempt to move the QP to the Error state. The QP is most likely unusable and should be destroyed.

Figure 24 - Affiliated Asynchronous Errors with Terminate Codes

Figure 25 indicates errors that cannot be associated with a QP; the Asynchronous Event Record MUST contain the additional information as indicated in the table.

Error	Terminate Code	Action
Locally detected Catastrophic Errors		
CQ Operation Error – An error occurred on the CQ unrelated to a specific QP completion.	None	The Asynchronous Event Record includes the CQ handle. All completions on the CQ are in an undefined state. It may be necessary to destroy any QPs targeting the CQ and destroy the CQ.
Shared Receive Queue Catastrophic Failure - A problem occurred with the RNIC or its driver that renders the RNIC unable to use the S-RQ.	None	The Asynchronous Event Record includes the S-RQ handle. All WRs on the S-RQ are in an undefined state. It may be necessary to destroy any QPs using the S_RQ and destroy the S-RQ.
RNIC Catastrophic failure - A problem occurred with the RNIC or its driver that renders the RNIC unable to reliably function. All RNIC/QP/CQ state is indeterminate. The only recovery is to close the RNIC (and reopen it if desired).	0x0208	The Asynchronous Event Record does not include any additional information. If possible, the RI Terminates all LLP Connections with Global Catastrophic Error. See 6.6.2.5

Figure 25 - Unaffiliated Asynchronous Errors with Terminate Codes

9 RNIC Verbs

The Verbs described in this chapter provide an abstract definition of the functionality provided to a host by a RI. Host RIs that are compliant with this specification MUST exhibit the semantic behavior described by the Verbs.

Since the Verbs define the behavior of the host RI, they may influence the design of software constructs, such as application programming interfaces (APIs), which provide access to the host RI. However, this specification explicitly does not define any such API. In particular, there is no requirement that an API used with a compliant host RI be semantically identical to, or expose the semantics of, the Verbs. For example, whether the input modifiers referenced in the Verbs are pass-by-reference or pass-by-value is outside the scope of this specification.

It is OPTIONAL for an RI to implement Block Lists. It is OPTIONAL for an RI to implement S-RQs. Support for S-RQs can be discovered using Query RNIC. Support for Block Lists can be discovered by attempting to open the RNIC in Block Mode. If the Verb fails with the error "Block List Not Supported", the RNIC does not support Block Mode.

The RI MUST use the values and information provided in the Input Modifiers when processing the requests and operations instantiated in the Verbs for mandatory features. The RI MUST use the values and information provided in the Input Modifiers when processing the requests and operations instantiated in the Verbs for optional features if the RI supports that optional feature.

9.1 Consumer Accessibility

Verb Consumers are the direct users of the Verbs, and are sub-divided into two classes, Privileged and Non-Privileged.

Privileged Consumers are typically those Consumers that operate at a privilege level sufficient to access OS internal data structures directly, and have the responsibility to control access to the RNIC Interface. All Verbs are available for use by Privileged Consumers.

Non-Privileged Mode Consumers are those Consumers that must rely on another agent, having a sufficient high level of privilege, to manipulate OS data structures. Only those Verbs specifically labeled as such are available to be used by Non-Privileged Mode Consumers. Conceptually, the intent is that Non-Privileged Mode Consumers are not allowed to manipulate RI resources that could affect a QP in a different Protection Domain. Any manipulation of resources that can affect another Protection Domain, such as registering physical

memory, are assumed to be done by a trusted intermediary, or Privileged Consumer.

The Protection Domain provides a mechanism to detect when a Consumer is posting WRs to QPs with which it is not associated. The RI also usually provides a mechanism to help prevent posting WRs to QPs not directly owned by the Consumer (e.g. a multi-Consumer application which shares the same PD). But it may still be possible to post a WR to a QP that is not owned by the Consumer in some environments. Preventing access to memory structures such as QPs not directly created by that Consumer can be partially provided by the Local Host's operating environment through the use of the virtual memory subsystem and mapping of RNIC resources. Since this is implementation and environment dependent, the mechanism describing it is outside the scope of the architecture.

All Verbs can be accessed by Privileged Mode Consumers. To maintain the access control over RI resources, the host environment MUST provide Non-Privileged Mode Consumers with direct access to only the following Verbs:

* PostSQ

* PostRQ

* Poll for Completion

* Request Completion Notification

9.2 RNIC Resource Management

9.2.1 RNIC

9.2.1.1 Open RNIC

Description:

Opens the specified RNIC.

The RI MUST support this Verb and MUST support all of the Input & Output Modifiers, except where noted. For more information, see Section 5.1.2 - Opening an RNIC.

Input Modifiers:

* The unique identifier for this RNIC. The naming scheme is implementation dependent.

* The Physical Block List mode of the RNIC. This MUST either be Block List mode or Page List mode. Block List mode is only valid if the RNIC supports it.

Output Modifiers:

* If the operation completed successfully:
    - RNIC Handle.
* Verb Results:
    - Operation completed successfully.
    - Insufficient resources to complete request.
    - Invalid Modifier (RNIC name).
    - Block List mode not supported.
    - RNIC in use.

9.2.1.2 Query RNIC

Description:

Returns the attributes for the specified RNIC.

The RI MUST support this Verb and MUST support all of the Input & Output Modifiers, except where noted. For more information, see Section 5.1.3 - Query RNIC.

Input Modifiers:

* RNIC handle.

Output Modifiers:

* RNIC Attributes & Values, if the operation completed successfully:
    - Vendor specific information. This could, but is not required to, include information such as a vendor identifier, part number and/or hardware version.
    - The maximum number of QPs supported by this RNIC.
    - The maximum number of outstanding Work Requests on any Send Queue or Receive Queue supported by this RNIC.

- The maximum number of outstanding Work Requests on any S-RQ supported by this RNIC. If S-RQs are not supported by this RNIC, this number is zero.
- The maximum number of Scatter/Gather Elements per Send Operation Type Work Request supported by this RNIC. This value also applies to the maximum number of Scatter/Gather Elements for WRs posted to Receive Queues as well as those posted to Shared-Receive Queues.
- The maximum number of Scatter/Gather Elements per RDMA Write Work Request supported by this RNIC.
- The maximum number of CQs supported by this RNIC.
- The maximum number of entries in each CQ supported by this RNIC.
- The maximum number of CQ Event Handlers supported by this RNIC.
- The maximum number of Memory Regions supported by this RNIC.
- The maximum number of Physical Buffer Entries per Physical Buffer List.
- The maximum number of Protection Domains supported by this RNIC.
- The maximum number of inbound RDMA Read Request Messages that can be in the IRRQ per RNIC. This is the per RNIC parameter that represents the maximum total value of IRD for all QPs. This value MUST be Zero if the resources used to handle Inbound RDMA Read Requests are not shared between QPs. (For more information, see Section 6.5 - Outstanding RDMA Read Resource Management)
- The maximum number of outbound RDMA Read Request Messages that can be outstanding per RNIC. This is the per RNIC parameter that represents the maximum total value of ORD for all QPs. This value is Zero if the resources used to handle outstanding Outbound RDMA Read Request Messages are not shared between QPs.
- The maximum number of inbound RDMA Read Request Messages that can be in the IRRQ per QP. This represents the maximum value for IRD for any QP.

- The maximum number of outbound RDMA Read Request Messages that can be outstanding per QP. This represents the maximum value for ORD for any QP.
- Ability of this RNIC to support modifying IRD after the QP has been created.
- Ability of this RNIC to support increasing ORD after the QP has been created.
- The maximum number of Memory Windows supported by this RNIC.
- The ability of this RNIC to support modifying the maximum number of outstanding Work Requests per QP. (For more information, see Section 6.1.3 - Modifying Queue Pair Attributes)
- The Physical Block List mode of the RNIC. This MUST either be Block List Mode or Page List Mode.
- If Block List Mode is supported:
  - The Physical Buffer Entry range of sizes supported by this RNIC.
- If Page List Mode is supported:
  - The List of Page sizes supported by this RNIC.
- The ability of this RNIC to support Shared Receive Queues.
- The ability of this RNIC to perform CQ Overflow detection.
- If Shared Receive Queues are supported:
  - The maximum number of Shared Receive Queues supported by this RNIC.
  - The dequeuing model the RNIC supports: arrival order or sequential order.

* Verb Results:
  - Operation completed successfully.
  - Invalid RNIC handle.

9.2.1.3 Close RNIC

Description:

Closes and resets the specified RNIC.

This Verb is responsible for de-allocating resources allocated by the RI and to make the RNIC unavailable for use by the Consumer.

The RI MUST support this Verb and MUST support all of the Input & Output Modifiers, except where noted. For more information, see Section 5.1.4 - Closing an RNIC.

Input Modifiers:

* RNIC handle.

Output Modifiers:

* Verb Results

  o Operation completed successfully.

  o Invalid RNIC handle.

9.2.2 Protection Domain

9.2.2.1 Allocate PD

Description:

Allocates an unused Protection Domain.

The RI MUST support this Verb and MUST support all of the Input & Output Modifiers, except where noted. For more information, see Section 5.2.1 - Allocating a PD.

Input Modifiers:

* RNIC Handle.

Output Modifiers:

* If the operation completed successfully:

  o PD ID.

* Verb Results:

  o Operation completed successfully.

  o Invalid RNIC handle.

- o Insufficient resources to complete request.

9.2.2.2 Deallocate PD

Description:

Deallocates a previously Allocated Protection Domain.

The RI MUST support this Verb and MUST support all of the Input & Output Modifiers, except where noted. For more information, see Section 5.2.2 - Deallocating a PD.

The Protection Domain MUST NOT be deallocated if it is still associated with any Queue Pair, Non-Shared Memory Region, Shared Memory Region, Shared Receive Queue, Bound Memory Window or Invalidated Memory Window.

Input Modifiers:

* RNIC Handle.
* PD ID.

Output Modifiers:

* Verb Results:
  - o Operation completed successfully.
  - o Invalid PD ID.
  - o Invalid RNIC handle.
  - o Protection Domain is in use.

9.2.3 Completion Queue

9.2.3.1 Create CQ

Description:

Creates a CQ on the specified RNIC. In addition, a Completion Event Handler may be registered for the created CQ.

The Consumer must specify the minimum number of entries in the CQ. The number of allocated entries for CQEs on the specified CQ, which might be different than the number requested, is returned on successful creation. The number returned differs only when the number of actual entries is greater than the number that the Consumer requested. If the maximum number of

entries the RNIC supports is less than the Consumer requested, an Immediate Error is returned and the CQ is not created.

The RI MUST support this Verb and MUST support all of the Input & Output Modifiers, except where noted. For more information, see Section 5.3.1 - Creating a Completion Queue.

Input Modifiers:

* RNIC handle.

* The minimum number of entries in the CQ.

* Completion Event Handler Identifier – An opaque handle used to identify a Completion Event Handler. If the identifier is set to zero, then there is no Completion Event Handler associated with this CQ. Completion Event Handler Identifiers are obtained via the Set Completion Event Handler Verb.

Output Modifiers:

* If the operation completed successfully:
  - The handle of the newly created CQ.
  - The allocated number of entries in the CQ.

* Verb Results:
  - Operation completed successfully.
  - Insufficient resources to complete request.
  - Invalid RNIC handle.
  - Number of CQ entries requested exceeds RNIC capability.
  - Invalid Completion Event Handler Identifier

9.2.3.2 Query CQ

Description:

Returns the number of entries in the specified CQ.

The RI MUST support this Verb and MUST support all of the Input & Output Modifiers, except where noted. For more information, see Section 5.3.2 - Querying Completion Queue Attributes.

Input Modifiers:

* RNIC handle.

* CQ handle.

Output Modifiers:

* If the operation completed successfully:

    o The allocated number of entries in the CQ.

    o The Completion Event Handler Identifier.

* Verb Results:

    o Operation completed successfully.

    o Invalid RNIC handle.

    o Invalid CQ handle.

9.2.3.3 Modify CQ

Description:

Resizes the CQ.

A CQ must be able to be resized with outstanding Work Completions on the CQ and Work Requests on queues associated with the specified CQ. If the requested minimum number of entries in the CQ is insufficient to hold the current number of entries on the CQ, an Immediate Error will result.

The RI MUST support this Verb and MUST support all of the Input & Output Modifiers, except where noted. For more information, see Section 5.3.3 - Modifying Completion Queue Attributes.

Input Modifiers:

* RNIC handle.

* CQ handle.

* The minimum number of entries in the CQ.

Output Modifiers:

* If the operation completed successfully:

    o The allocated number of entries in the CQ.

* Verb Results:
    - Operation completed successfully.
    - Insufficient resources to complete request.
    - Invalid RNIC handle.
    - Invalid CQ handle.
    - Number of CQ entries requested exceeds RNIC capability.
    - An Attempt to shrink the size of the queue failed because too many Completion Queue Entries were still present on the Completion Queue.

9.2.3.4 Destroy CQ

Description:

Destroys the specified CQ.

The CQ cannot be destroyed if any Work Queue is still associated with the CQ.

The RI MUST support this Verb and MUST support all of the Input & Output Modifiers, except where noted. For more information, see Section 5.3.4 - Destroying a Completion Queue.

Input Modifiers:

* RNIC handle.
* CQ handle.

Output Modifiers:

* Verb Results:
    - Operation completed successfully.
    - Invalid RNIC handle.
    - Invalid CQ handle.
    - One or more Work Queues is still associated with the CQ.

9.2.4 Shared Receive Queue

9.2.4.1 Create S-RQ

Description:

Creates an S-RQ for the specified RNIC.

A set of initial S-RQ attributes must be specified by the Consumer. If any of the required initial attributes are illegal or missing, an error is returned and the S-RQ is not created.

The RI MUST support this Verb if the Query RNIC Output Modifier indicates support for an S-RQ and MUST support all of the Input & Output Modifiers in this case, except where noted. For more information, see Section 6.3.1 - Creating a Shared Receive Queue.

Input Modifiers:

* RNIC handle.
* The maximum number of outstanding Work Requests the Consumer expects to submit to the Shared Receive Queue.
* The S-RQ Limit. The S-RQ Limit detection is armed by the RI upon creation of the S-RQ, if the S-RQ Limit is non-zero.
* The maximum number of Scatter/Gather Elements the Consumer can specify in a Work Request.
* PD ID.

Output Modifiers:

* If the operation completed successfully:
  - The S-RQ Handle.
  - The allocated number of outstanding Work Requests the Consumer can submit to the Shared Receive Queue.
  - The allocated number of scatter/gather elements that can be specified in Work Requests. If an error is not returned, this is guaranteed to be greater than or equal to the number requested.
* Verb Results:
  - Operation completed successfully.

- Insufficient resources to complete request.
- Invalid RNIC handle.
- Maximum number of Work Requests requested exceeds RNIC capability.
- Maximum number of scatter/gather elements per Receive Queue Work Request requested exceeds RNIC capability.
- Invalid PD ID.
- S-RQ Limit out of range.

9.2.4.2 Query S-RQ

Description:

Returns the attribute list and current values for the specified S-RQ.

The RI MUST support this Verb if the Query RNIC Output Modifier indicates support for an S-RQ and MUST support all of the Input & Output Modifiers in this case, except where noted.

Input Modifiers:

* RNIC Handle.
* S-RQ Handle.

Output Modifiers:

* The S-RQ attributes, if the operation completed successfully. The list of attributes returned by the query are:
  - The allocated number of outstanding Work Requests supported on the Shared Receive Queue.
  - The allocated number of Scatter/Gather Elements supported on Work Requests submitted to the Shared Receive Queue.
  - PD ID.
  - The S-RQ Limit.
  - S-RQ Limit Armed Indicator.
* Verb Results:

- Operation completed successfully.
- Invalid RNIC handle.
- Invalid S-RQ handle.

9.2.4.3 Modify S-RQ

Description:

Modifies the attributes for the specified S-RQ.

The RI MUST support this Verb if the Query RNIC Output Modifier indicates support for an S-RQ and MUST support all of the Input & Output Modifiers in this case, except where noted. For more information, see Section 6.3.2 - Modifying a Shared Receive Queue.

Input Modifiers:

* RNIC Handle.
* S-RQ Handle.
* The S-RQ attributes to modify and their new values. The S-RQ attributes that can be modified after the S-RQ has been created are:
  - The maximum number of outstanding Work Requests the Consumer expects to submit to the Shared Receive Queue (if changing is supported by the RNIC).
  - The S-RQ Limit.
  - Re-arm the S-RQ Limit Asynchronous Event.

Output Modifiers:

* If the operation completed successfully:
  - The allocated number of outstanding Work Requests supported on the Shared Receive Queue.
* Verb Results:
  - Operation completed successfully.
  - Insufficient resources to complete request.
  - Invalid RNIC handle.

- Invalid S-RQ handle.
- Maximum number of Shared Receive Queue Work Requests requested exceeds RNIC capability.
- An Attempt to shrink the size of the queue failed because too many elements were still present.
- S-RQ Limit out of range.
- Invalid Input Modifier.

9.2.4.4 Destroy S-RQ

Description:

Destroys the specified S-RQ.

The RI MUST support this Verb if the Query RNIC Output Modifier indicates support for an S-RQ and MUST support all of the Input & Output Modifiers in this case, except where noted.

For more information, see Section 6.3.3 - Destroying a Shared Receive Queue.

Input Modifiers:

* RNIC handle.
* S-RQ handle.

Output Modifiers:

* Verb Results:
  - Operation completed successfully.
  - Invalid RNIC handle.
  - Invalid S-RQ handle.
  - QPs still associated with the S-RQ.

9.2.5 Queue Pair

9.2.5.1 Create QP

Description:

Creates a QP for the specified RNIC.

A set of initial QP attributes must be specified by the Consumer. If any of the required initial attributes are illegal or missing, an error is returned and the Queue Pair is not created.

The RI MUST support this Verb and MUST support all of the Input & Output Modifiers, except where noted. For more information, see Section 6.1.1 - Creating a Queue Pair.

Input Modifiers:

* RNIC handle.

* The QP attributes that must be specified at QP create time are:

  - The CQ handle of the CQ to be associated with the Send Queue.

  - The CQ handle of the CQ to be associated with the Receive Queue. (Note that this may be the same CQ that is associated with the Send Queue, or it may be a different CQ than the one associated with the Send Queue).

  - The maximum number of outstanding Work Requests the Consumer expects to submit to the Send Queue.

  - The maximum number of outstanding Work Requests the Consumer expects to submit to the Receive Queue. This value is ignored if the QP is associated with an S-RQ.

  - If the QP's RQ will be associated with an S-RQ:

    + S-RQ Handle.

    + QP RQ Limit Indicator, as discussed in Section 6.3.8 - S-RQ Limit Checking. The QP RQ Limit detection is armed by the RI upon creation of the QP, if non-zero.

  - Inbound RDMA Read enable.

  - Inbound RDMA Write and inbound RDMA Read Response enable.

  - Bind Memory Windows enable.

  - The maximum number of scatter/gather elements the Consumer can specify in a Send Operation Type Work Request submitted to the Send Queue.

- The maximum number of scatter/gather elements the Consumer can specify in a RDMA Write Work Request submitted to the Send Queue.
- The maximum number of scatter/gather elements the Consumer can specify in a Work Request submitted to the Receive Queue. This value is not returned if the QP is associated with an S-RQ.
- ORD (Requested) - The requested maximum number of outstanding Outgoing RDMA Read Request Messages the RNIC can initiate from the SQ.
- IRD (Requested) - The requested maximum number of outstanding Incoming RDMA Read Request Messages (e.g. IRRQ depth) the RNIC can handle for this QP.
- PD ID.
- Enable or disable the Use of the STag of zero and Fast-Register Non-Shared Memory Region Operations. This MUST only be allowed to be enabled for Privileged Mode Consumers.

Output Modifiers:

* If the operation completed successfully:
  - The QP Handle.
  - The QP ID.
  - The allocated number of outstanding Work Requests supported on the Send Queue. If an error is not returned, this is guaranteed to be greater than or equal to the number requested. (This may require the Consumer to increase the size of the CQ.)
  - The allocated number of outstanding Work Requests supported on the Receive Queue. If an error is not returned, this is guaranteed to be greater than or equal to the number requested. (This may require the Consumer to increase the size of the CQ.) This value is not returned if the QP is associated with an S-RQ.
  - The allocated number of scatter/gather elements that can be specified in Work Requests submitted to the Send Queue. If an error is not returned, this is guaranteed to be greater than or equal to the number requested.

- The allocated number of Scatter/Gather Elements supported on RDMA Write Work Requests submitted to the Send Queue. If an error is not returned, this is guaranteed to be greater than or equal to the number requested.

- The allocated number of Scatter/Gather Elements that can be specified in Work Requests submitted to the Receive Queue. If an error is not returned, this is guaranteed to be greater than or equal to the number requested. This value is not returned if the QP is associated with an S-RQ.

- ORD (allocated) - The allocated number of outstanding RDMA Read Request Messages the RNIC can initiate from the SQ at the Data Sink. This number MUST be between zero and the number requested, inclusive. If the Consumer requested a non-zero number and the RI was unable to provision at least one then an Immediate Error MUST be returned.

- IRD (allocated) - The allocated number of incoming outstanding RDMA Read Request Messages (e.g. IRRQ depth) the RNIC's QP can handle at the Data Source. If the Consumer requested a non-zero number and the RI was unable to provision at least one then an Immediate Error MUST be returned.

* Verb Results:

  - Operation completed successfully.

  - Insufficient resources to complete request.

  - Invalid RNIC handle.

  - Invalid CQ handle.

  - Invalid S-RQ handle.

  - The value requested for ORD exceeds RNIC capability.

  - The value requested for IRD exceeds RNIC capability.

  - Maximum number of Send Queue Work Requests requested exceeds RNIC capability.

  - Maximum number of Receive Queue Work Requests requested exceeds RNIC capability

  - Maximum number of scatter/gather elements per Send Queue Work Request requested exceeds RNIC capability.

- Maximum number of scatter/gather elements per Receive Queue Work Request requested exceeds RNIC capability.
- Invalid Protection Domain.
- QP RQ Limit Out of Range.

9.2.5.2 Query QP

Description:

Returns the attribute list and current values for the specified QP.

The RI MUST support this Verb and MUST support all of the Input & Output Modifiers, except where noted. For more information, see Section 6.1.2 - Querying Queue Pair Attributes.

Input Modifiers:

* RNIC Handle.
* QP Handle.

Output Modifiers:

* The QP attributes, if the operation completed successfully. The list of attributes returned by the query are:
  - Handle of the Completion Queue associated with the Send Queue.
  - Handle of the Completion Queue associated with the Receive Queue.
  - Handle of the S-RQ. This value is only returned if the QP is associated with an S-RQ.
  - The allocated number of outstanding Work Requests supported on the Send Queue.
  - The allocated number of outstanding Work Requests supported on the Receive Queue. This value is not returned if the QP is associated with an S-RQ.
  - The actual number of Scatter/Gather Elements supported on Send Operation Type Work Requests submitted to the Send Queue.

- The allocated number of Scatter/Gather Elements supported on RDMA Write Work Requests submitted to the Send Queue.
- The allocated number of Scatter/Gather Elements supported on Work Requests submitted to the Receive Queue. This value is not returned if the QP is associated with an S-RQ.
- ORD - The allocated number of outstanding RDMA Read Request Messages the RNIC can initiate from the SQ at the Data Sink.
- IRD - The allocated number of outstanding incoming RDMA Read Request Messages (e.g. IRRQ depth) the RNIC's QP can handle at the Data Source.
- Current QP state.
- PD ID.
- QP ID.
- Use of the STag of zero and Fast-Register Non-Shared Memory Region Operations enabled.
- Inbound RDMA Read enable.
- Inbound RDMA Write and inbound RDMA Read Response enable.
- Bind Memory Windows enable.

The following attributes are not defined unless the QP is in the Terminate or Error states.

- A buffer containing the Terminate Message that was received or sent (if possible).
- An indicator to state if the Terminate Message was generated locally or by the Associated QP.

The following attributes are only defined if the QP is associated with a Shared Receive Queue.

- Current QP's RQ Limit.
- QP's RQ Limit armed indicator.

The following attributes are only defined if the QP is not in the Idle state.

- LLP Stream Handle.

* Verb Results:
    - Operation completed successfully.
    - Invalid RNIC handle.
    - Invalid QP handle.

9.2.5.3 Modify QP

Description:

Modifies the attributes for the specified QP then causes the QP to transition to the specified QP state. Only a subset of the QP attributes can be modified in each of the QP states.

The RI MUST support this Verb and MUST support all of the Input & Output Modifiers, except where noted. For more information, see Section 6.1.3 - Modifying Queue Pair Attributes.

Input Modifiers:

* RNIC Handle.
* QP Handle.
* The QP attributes to modify and their new values. The QP attributes that can be modified after the QP has been created are:
    - Next QP state. If the current state is specified, only the QP attributes will be modified.
    - ORD - The requested number of outstanding RDMA Read Request Messages the RNIC can initiate from the SQ at the Data Sink.
    - IRD - The requested number of incoming outstanding RDMA Read Request Messages (e.g. IRRQ depth) the RNIC's QP can handle at the Data Source.
    - The maximum number of outstanding Work Requests the Consumer expects to submit to the Send Queue (if changing is supported by the RNIC).
    - The maximum number of outstanding Work Requests the Consumer expects to submit to the Receive Queue (if changing is supported by the RNIC). This value is not allowed if the QP is associated with an S-RQ.

The following attributes are only defined if the QP is associated with a Shared Receive Queue.

- QP's RQ Limit, as described in Section 6.3.8 - S-RQ Limit Checking.
- Re-arm the QP's RQ Limit, as described in Section 6.3.8 - S-RQ Limit Checking. The RI MUST allow an already armed S-RQ limit to be armed.

Valid only when moving from Idle to RTS.

- LLP Stream Handle
- Stream Message Buffer.

Output Modifiers:

* If the operation completed successfully:
  - The allocated number of outstanding Work Requests supported on the Send Queue.
  - The allocated number of outstanding Work Requests supported on the Receive Queue. This value is not returned if the QP is associated with an S-RQ.
  - ORD - The allocated number of outstanding RDMA Read Request Messages the RNIC can initiate from the SQ at the Data Sink. This number MUST be between zero and the number requested, inclusive. If the Consumer requested a non-zero number and was unable to provision at least one then an Immediate Error will be returned.
  - IRD - The allocated number of incoming outstanding RDMA Read Request Messages (e.g. the IRRQ depth) the RNIC’s QP can handle at the Data Source. If the Consumer requested a non-zero number and was unable to provision at least one then an Immediate Error will be returned.
* Verb Results:
  - Operation completed successfully.
  - Insufficient resources to complete request.
  - Invalid RNIC handle.
  - Invalid QP handle.

- Cannot change QP attribute.
- Invalid QP state change requested.
- Maximum number of Send Queue Work Requests requested exceeds RNIC capability.
- Maximum number of Receive Queue Work Requests requested exceeds RNIC capability.
- The value requested for ORD exceeds RNIC capability.
- The value requested for IRD exceeds RNIC capability.
- An Attempt to shrink the size of the queue failed because too many elements were still present.
- Invalid LLP Stream Handle.
- Invalid Modifier.
- RI still flushing WQEs.
- RQ Limit Out of Range.

9.2.5.4 Destroy QP

Description:

Destroys the specified QP.

The RI MUST support this Verb and MUST support all of the Input & Output Modifiers, except where noted. The QP cannot be destroyed if any Memory Windows are still Bound to the QP.

For more information, see Section 6.1.4 - Destroying a Queue Pair.

Input Modifiers:

* RNIC handle.
* QP handle.

Output Modifiers:

* Verb Results:
  - Operation completed successfully.

- Invalid RNIC handle.
- Invalid QP handle.
- Memory Windows still Bound to QP.

9.2.6 Memory Management

Memory Management Verbs are used to manage Memory Regions and Memory Windows. The following table describes what each of the Memory Management Verbs manage and where the Verb appears to performed:

Verb	Used to manage MR vs. MW	Performed by RI vs. RNIC
Allocate Non-Shared Memory Region STag	MR	RI
Register Non-Shared Memory Region (RI-Register)	MR	RI
Reregister Non-Shared Memory Region (RI-Reregister)	MR	RI
Register Shared Memory Region	MR	RI
Fast-Register Non-Shared Memory Region (PostSQ)	MR	RNIC
Query Memory Region	MR	RI
Invalidate Local STag (PostSQ)	MR or MW	RNIC
Deallocate STag	MR or MW	RI
Allocate Memory Window	MW	RI
Query Memory Window	MW	RI
Bind Memory Window (PostSQ)	MW	RNIC

Figure 26 - Memory Management Verbs

9.2.6.1 Allocate Non-Shared Memory Region STag

Description:

Allocates memory registration resources on the RNIC.

The RI MUST support this Verb and MUST support all of the Input & Output Modifiers, except where noted. For more information, see Section 7.3.2.1 - Allocate Non-Shared Memory Region STag.

Input Modifiers:

* RNIC Handle.

* Requested Physical Buffer List size to be allocated.

* PD ID.

* Remote Access Flag. If set, Local and Remote Access is enabled. Otherwise only Local access is enabled.

Output Modifiers:

* If the operation completed successfully:

  o STag Index - used for local and, if specified by the input modifiers, remote access.

  o The actual number of Physical Buffer List Entries in the allocated Physical Buffer List. Note that this MAY be greater than the number requested.

* Verb Results:

  o Operation completed successfully.

  o Insufficient resources to complete request.

  o Invalid RNIC handle.

  o Invalid PD ID.

9.2.6.2 Register Non-Shared Memory Region (RI-Register)

Description:

Registers a Non-Shared Memory Region for use by an RNIC.

The RI MUST support this Verb and MUST support all of the Input & Output Modifiers, except where noted. For more information, see Section 7.3.2.2 - RI-Register Non-Shared Memory Region.

Input Modifiers:

* RNIC Handle.

* Physical Buffer Entry size - The size, in bytes, of each Physical Buffer in the list. Note: If the Physical Buffer List references a Page List, the size MUST be a power of two. If the Physical Buffer List references a Block List, the size MAY have a byte alignment.

* Address List - A list of addresses that point to the Physical Buffers referenced by the Physical Buffer List. All Physical Buffers in the list have the same size.

* Address List Length - the number of entries in the Address list.

* First Byte Offset (FBO) - Offset to start of Non-Shared Memory Region on first Physical Buffer.

* Length - Total length of the Non-Shared Memory Region (can be of arbitrary byte-aligned length).

* Addressing type. The Addressing type MUST be one of the following:

    o VA Based TO

    o Zero Based TO

* The following input modifier is only valid if the Addressing type is VA Based TO:

    o Virtual Address – The VA address of the first byte in the Non-Shared Memory Region.

* PD ID.

* STag Key.

* Remote Access Flag.

* Access Control - The following MAY be selected in any combination except as noted:

    o Enable Local Write Access.

    o Enable Remote Write Access. Remote Write Access requires Local Write Access to be enabled.

    o Enable Local Read Access.

    o Enable Remote Read Access. Remote Read Access requires Local Read Access to be enabled.

- Enable Memory Window Binding.

Output Modifiers:

* If the operation completed successfully:
  - STag Index - used for local and, if specified by the input modifiers, remote access. Note: the RNIC associates the STag Key passed in as an input modifier to STag associated with the registered Non-Shared Memory Region.
  - The actual number of Physical Buffer List Entries in the allocated Physical Buffer List. Note that this MAY be greater than the number requested.
* Verb Results:
  - Operation completed successfully.
  - Insufficient resources to complete request.
  - Invalid RNIC handle.
  - Invalid PD ID.
  - Invalid Virtual Address.
  - Invalid length.
  - Invalid First Byte Offset.
  - Invalid Access Rights requested.
  - Invalid Physical Buffer List entry.
  - Invalid Physical Buffer size.

9.2.6.3 Query Memory Region

Description:

Retrieves information about a specific Memory Region.

The RI MUST support this Verb and MUST support all of the Input & Output Modifiers, except where noted. For more information, see Section 7.7 - Querying Memory Regions.

Input Modifiers:

* RNIC Handle.

* STag Index - as originally returned from an Allocate Non-Shared Memory Region STag, RI-Register Non-Shared Memory Region, RI-Reregister Non-Shared Memory Region or Register Shared Memory Region Type Verb.

Output Modifiers:

* If the operation completed successfully:
    - STag Key - Current STag Key associated with the Memory Region, if it is in the Valid state.
    - Remote Access Flag.
    - PD ID.
    - STag State: Valid or Invalid.
    - STag Type: Shared or Non-Shared.
    - The actual number of Physical Buffer List Entries in the allocated Physical Buffer List. Note that this MAY be greater than the number requested.
    - Access Control settings for the registered Region. The following MAY be set in any combination except as noted:
        + Local Write Access Enabled.
        + Remote Write Access Enabled. Remote Write Access requires Local Write Access to be enabled.
        + Local Read Access Enabled.
        + Remote Read Access Enabled. Remote Read Access requires Local Read Access to be enabled.
        + Memory Window Binding Enabled.
* Verb Results:
    - Operation completed successfully.
    - Invalid RNIC handle.
    - Invalid STag Index.

9.2.6.4 Deallocate STag

Description:

Removes an STag created through an Allocate Non-Shared Memory Region STag, RI-Register Non-Shared Memory Region, RI-Reregister Non-Shared Memory Region, Register Shared Memory Region or Allocate Memory Window from the RNIC.

Work Requests or Remote Operation requests that are in-process and actively referencing memory locations associated with the STag being deallocated must fail with a protection error.

If the STag references a Memory Region which has Memory Windows Bound to it, an immediate Error MUST be returned and the Memory Region must not be destroyed or modified.

The RI MUST support this Verb and MUST support all of the Input & Output Modifiers, except where noted. For more information, see Section 7.9 - Deallocation of STag associated with a Memory Region and Section 7.10.4 - Invalidating or De-allocating Memory Windows.

Input Modifiers:

* RNIC Handle.

* STag Index - as originally returned from an Allocate Non-Shared Memory Region STag, Allocate Memory Window, or RI-Register Non-Shared Memory Region, RI-Reregister Non-Shared Memory Region or Register Shared Memory Region Verb.

Output Modifiers:

* Verb Results:

  - Operation completed successfully.

  - Invalid RNIC handle.

  - Invalid STag Index.

  - One or more Memory Windows is still Bound to the Memory Region. Applies only if the STag is associated with a Memory Region.

9.2.6.5 Reregister Non-Shared Memory Region (RI-Reregister)

Description:

Modifies the attributes of an existing Non-Shared Memory Region.

The STag output modifier from this Verb must be used in place of any previously issued for this Non-Shared Memory Region.

If the STag references a Non-Shared Memory Region which has Memory Windows Bound to it, an immediate Error MUST be returned and the Non-Shared Memory Region must not be destroyed or modified.

The RI MUST support this Verb and MUST support all of the Input & Output Modifiers, except where noted. For more information, see Section 7.3.2.3 - RI-Reregister Non-Shared Memory Region.

Input Modifiers:

* RNIC Handle.

* Physical Buffer Entry size - The size, in bytes, of each Physical Buffer Entry in the list. Note: If the Physical Buffer List references a Page-List, the size MUST be a power of two. If the Physical Buffer List references a Block-List, the size MAY have a byte alignment.

* Address List - A list of addresses that point to the Physical Buffers referenced by the Physical Buffer List. All Physical Buffers in the list MUST have the same size.

* Address List Length - the number of entries in the Address list.

* First Byte Offset (FBO) - Offset to start of Non-Shared Memory Region on first Physical Buffer.

* Length - Total length of Non-Shared Memory Region (can be of arbitrary byte-aligned length).

* Addressing type. The addressing type MUST be one of the following:

  o VA Based TO

  o Zero Based TO

* The following input modifier is only valid if the Addressing type is VA Based TO:

  o Virtual Address – The VA address of the first byte in the Non-Shared Memory Region.

* PD ID.

* STag Index.

* STag Key (not the existing STag Key, but the new STag Key).

* Remote Access Flag.

* Access Control - The following MAY be selected in any combination except as noted:

    - Enable Local Write Access.

    - Enable Remote Write Access. Remote Write Access requires Local Write Access to be enabled.

    - Enable Local Read Access.

    - Enable Remote Read Access. Remote Read Access requires Local Read Access to be enabled.

    - Enable Memory Window Binding.

Output Modifiers:

* If the operation completed successfully:

    - STag Index - used for local and, if specified by the input modifiers, remote access. Note: the RNIC associates the STag Key passed in as an input modifier to STag associated with the registered Non-Shared Memory Region. If the output STag index differs from the input STag index, the old STag index was Deallocated.

    - The actual number of Physical Buffer List Entries in the allocated Physical Buffer List. Note that this MAY be greater than the number requested.

* Verb Results:

    - Operation completed successfully.

    - Insufficient resources to complete request.

    - Invalid RNIC handle.

    - Invalid STag Index.

    - Invalid Virtual Address.

    - Invalid Length.

    - Invalid PD ID.

    - Invalid First Byte Offset.

- Invalid Access Rights request.
- One or more Memory Windows is still Bound to the Region.
- Invalid Physical Buffer List entry.
- Invalid Physical Buffer size.

9.2.6.6 Register Shared Memory Region

Description:

Registers a new Shared Memory Region which shares RNIC mapping resources with a previously registered Memory Region, thus returning a new STag. Note that other than the change of the original Memory Region to a Shared Memory Region, the original Memory Region remains unaffected by this operation.

The Base TO,VA (if the input STag Index references a VA Based TO), PD ID, and Access Rights specified for the new Memory Region need not be the same as those of the existing Memory Region. The lengths are by definition the same.

The RI MUST support this Verb and MUST support all of the Input & Output Modifiers, except where noted. For more information, see Section 7.4.3 - Multiple Registrations of Memory Regions.

Input Modifiers:

* RNIC Handle.
* STag Index of the existing Memory Region. If the existing Memory Region is Non-Shared, successful completion of this verb will convert the existing Non-Shared Memory Region to a Shared Memory Region.
* Addressing type. The addressing type MUST be one of the following:
  - VA Based TO
  - Zero Based TO
* The following modifier is only valid if the Addressing type of the existing region is VA Based TO:
  - Virtual Address – The VA address of the first byte in the Memory Region.
* PD ID.

* STag Key of the new STag.

* Remote Access Flag.

* Access Control - The following MAY be selected in any combination except as noted:

    - Enable Local Write Access.

    - Enable Remote Write Access. Remote Write Access requires Local Write Access to be enabled.

    - Enable Local Read Access.

    - Enable Remote Read Access. Remote Read Access requires Local Read Access to be enabled.

    - Enable Memory Window Binding.

Output Modifiers:

* If the operation completed successfully:

    - STag Index - used for local and, if specified by the input modifiers, remote access. Note: the RNIC associates the STag Key passed in as an input modifier to STag associated with the registered Shared Memory Region.

* Verb Results:

    - Operation completed successfully.

    - Insufficient resources to complete request.

    - Invalid RNIC handle.

    - Invalid STag Index.

    - Invalid Virtual Address.

    - Invalid PD ID.

    - Invalid Access Rights requested.

9.2.6.7 Allocate Memory Window

Description:

This Verb allocates a memory window and associates it with a Protection Domain. It is not inherently associated with any Memory Region when allocated.

The RI MUST support this Verb and MUST support all of the Input & Output Modifiers, except where noted. For more information, see Section 7.10.1 - Allocating Memory Windows.

Input Modifiers:

* RNIC Handle.

* PD ID.

Output Modifiers:

* If the operation completed successfully:

  o STag Index - an unbound STag for use in specifying the Window when invoking a Bind Work Request through the Post Send Verb.

* Verb Results:

  o Operation completed successfully.

  o Insufficient resources to complete request.

  o Invalid RNIC handle.

  o Invalid PD ID.

9.2.6.8 Query Memory Window

Description:

This Verb returns the attributes associated with the specified memory window.

The RI MUST support this Verb and MUST support all of the Input & Output Modifiers, except where noted. For more information, see Section 7.10.3 - Memory Windows.

Input Modifiers:

* RNIC Handle.

* STag Index - the current STag associated with the Memory Window.

Output Modifiers:

* If the operation completed successfully:
    - STag Key - current value of the STag Key, if the STag is in the Valid state.
    - STag State: Valid or Invalid.
    - PD ID.
    - Access Rights. The following may be set in any combination except as noted.
        + Remote Write Access Enabled. If set Remote Write Access is enabled.
        + Remote Read Access Enabled. If set Remote Read Access is enabled.
* Verb Results:
    - Operation completed successfully.
    - Invalid RNIC handle.
    - Invalid STag Index.

9.3 Work Request Processing

9.3.1 QP Operations

9.3.1.1 PostSQ

Description:

Builds a WQE on the Send Queue of the specified QP for each entry in the Work Request List submitted by the Consumer. This WQE is added to the end of the Send Queue and the RNIC is notified that a new WQE is ready to be processed.

Note that not all Input Modifiers are valid for all operations. If Input Modifiers are specified that are not valid for a particular operation, they are ignored.

Following the Verbs is a Work Request table which contains a List of the Operation Types and the Input Modifiers which are required for each of those Operation Types.

The RI MUST support this Verb and MUST support all of the Input & Output Modifiers, except where noted. For more information, see Section 8.2.1 - Submitting Work Request to a Work Queue.

Input Modifiers:

* RNIC Handle

* QP Handle.

* A list of Work Requests. Each Work Request MUST contain the following information:

  - A user defined 64-bit Work Request ID

  - Operation type. The operation type MUST be one of the following:

    + Send

    + Send with Solicited Event

    + Send with Invalidate

    + Send with Solicited Event & Invalidate

    + RDMA Write

    + RDMA Read

    + RDMA Read with Invalidate Local STag

    + Bind Memory Window

    + Fast-Register Non-Shared Memory Region

    + Invalidate Local STag

  - Completion Notification Type: Signaled or Unsignaled.

  - The following list of modifiers are only valid for Send Operation Types and RDMA Write WRs to represent the Local Buffer:

    + Scatter/Gather List. The Scatter/Gather List can contain zero or more Scatter/Gather Elements. This list is specified only for Send and RDMA type operations.

    + Number of Scatter/Gather Elements.

    + Note that the length is determined by adding up the Length field in the SGEs of the SGL.

    + Read Fence indicator.

- The following list of modifiers are only valid for RDMA Read Type operations to represent the Local Buffer:
  - Local Address. This is a contiguous buffer represented by a TO, an STag, and a Length to be read.
- The following list of modifiers are only valid for RDMA Write or RDMA Read Type WRs to represent the Remote Buffer:
  - Remote Address. This is a contiguous buffer represented by a TO and an STag.
- The following modifier is only valid for the Send with Invalidate and Send with Solicited Event & Invalidate operations:
  - Remote STag. This is the STag to be Invalidated at the Remote Peer.
- The following list of modifiers are only valid for Bind Memory Window operations:
  - STag Index for the Memory Window.
  - STag Key for the Memory Window.
  - STag for the Memory Region that the Memory Windows is to be associated with. This parameter includes both the STag Index and STag Key.
  - Length or range to be Bound in number of octets.
  - Addressing type. The addressing type MUST be one of the following:
    - VA Based TO
    - Zero Based TO
  - Virtual Address - The VA address of the first byte into the Memory Region. This may be different than the starting address of the Memory Region.
  - Access Control - either or both of the following must be selected:
    - Enable Remote Write Access. Requires the Memory Region to have Local Write Access.

        * Enable Remote Read Access. Requires the Memory Region to have Local Read Access.

- The following list of modifiers are only valid for Fast-Register Non-Shared Memory Region operations:
  - Physical Buffer Entry size - The size, in bytes, of each Physical Buffer in the list. Note: If the Physical Buffer List references a Page-List, the size MUST be a power of two. If the Physical Buffer List references a Block-List, the size MUST be an RNIC supported size (see Section 9.2.1.2 - Query RNIC).
  - Address List - A list of addresses that point to the Physical Buffers referenced by the Physical Buffer List. All Physical Buffers in the list MUST have the same size.
  - Address List Length - the number of entries in the Address list.
  - First Byte Offset (FBO) - Offset to start of Non-Shared Memory Region on first Physical Buffer.
  - Length - Total length of Non-Shared Memory Region (can be any value supported by the RNIC).
  - Addressing type. The addressing type MUST be one of the following:
    * VA Based TO
    * Zero Based TO
  - The following modifier is only valid if the Addressing type is VA Based TO:
    * Virtual Address – The VA address of the first byte in the Non-Shared Memory Region
  - STag Index.
  - STag Key.
  - Access Control - The following may be selected in any combination except as noted:
    * Enable Local Write Access.

    * Enable Remote Write Access. Remote Write Access requires Local Write Access to be enabled. The STag Index MUST have the Remote Access Flag enabled.

    * Enable Local Read Access.

    * Enable Remote Read Access. Remote Read Access requires Local Read Access to be enabled. The STag Index MUST have the Remote Access Flag enabled.

    * Enable Memory Window Binding.

- The following list of modifiers are only valid for Invalidate Local STag operations:

    + STag to be the target of the Invalidate operation.

    + Local Fence indicator.

Below, in Figure 27, is a matrix of the Input Modifiers for PostSQ and the Operation Types. The intersection of the matrix indicates that the Input Modifier is required for that Operation Type by specifying "Yes".

Opcode-> Input Modifier	**Send**	**Send w/ SE**	**Send w/ Inv.**	**Send w/ SE & Inv.**	**RDMA Write**	**RDMA Read**	**RDMA Read w/ Inv.**	**Bind MW**	**Fast-Reg. NS MR**	**Inv. Local STag**
WR ID	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Compltn. Notif. Type	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
SGL	Yes	Yes	Yes	Yes	Yes					
SGE No.	Yes	Yes	Yes	Yes	Yes					
Read Fence	Yes	Yes	Yes	Yes	Yes					
Local Fence										Yes
Local Address						Yes	Yes			

Opcode-> Input Modifier	**Send**	**Send w/ SE**	**Send w/ Inv.**	**Send w/ SE & Inv.**	**RDMA Write**	**RDMA Read**	**RDMA Read w/ Inv.**	**Bind MW**	**Fast-Reg. NS MR**	**Inv. Local STag**
Remote Address					Yes	Yes	Yes			
Remote STag			Yes	Yes						
MW STag Key								Yes		
MW STag Index								Yes		
MW's MR STag								Yes		
MW Length								Yes		
Addr Type								Yes	Yes	
VA, if VA Based TO								Yes	Yes	
Acs Ctrl: Local Rd									Yes	
Acs Ctrl: Remote Rd								Yes	Yes	
Acs Ctrl: Local Wt									Yes	
Acs Ctrl: Remote Wt								Yes	Yes	

Opcode-> Input Modifier	**Send**	**Send w/ SE**	**Send w/ Inv.**	**Send w/ SE & Inv.**	**RDMA Write**	**RDMA Read**	**RDMA Read w/ Inv.**	**Bind MW**	**Fast-Reg. NS MR**	**Inv. Local STag**
Acs Ctrl: Bind Enable									Yes	
PBLE Size									Yes	
PBL									Yes	
FBO									Yes	
STag Index									Yes	Yes
STag Key									Yes	Yes

Figure 27 - PostSQ Input Modifier Validity

Output Modifiers:

* Number of WRs posted.
* Verb Results:
    - Operation completed successfully
    - Invalid RNIC Handle
    - Invalid QP Handle
    - Too many Work Requests posted.
    - Invalid operation type.
    - Invalid QP state.
    - Invalid Scatter/Gather list format.
    - Invalid Scatter/Gather list length.
    - Invalid Modifier.

9.3.1.2 PostRQ

Description:

Builds a WQE on the Receive Queue of the specified QP for each entry in the Work Request List submitted by the Consumer. This WQE is added to the end of the Receive Queue and the RNIC is notified that a new WQE is ready to be processed.

The RI MUST support this Verb and MUST support all of the Input & Output Modifiers, except where noted. For more information, see Section 8.2.1 - Submitting Work Request to a Work Queue.

Input Modifiers:

* RNIC Handle.
* QP Handle, for QP's not associated with an S-RQ.
* S-RQ Handle, for QP's associated with an S-RQ.
* A list of Work Requests. Each Work Request MUST contain the following information.
  - A user defined 64-bit Work Request ID.
  - Scatter/Gather List. The scatter/gather list can contain one or more Data Segments.
  - Number of Scatter/Gather List elements.

Output Modifiers:

* Number of WRs posted.
* Verb Results:
  - Operation completed successfully.
  - Invalid RNIC handle.
  - Invalid QP handle.
  - Invalid S-RQ handle.
  - Too many Work Requests posted.
  - Invalid QP state.
  - Invalid Scatter/Gather list format.

- o Invalid Scatter/Gather list length.
- o Invalid Modifier.
- o RQ Associated with S-RQ.

9.3.2 CQ Operations

9.3.2.1 Poll for Completion (Poll CQ)

Description:

Polls the specified CQ for a Work Completion.

If a CQE is present, the CQE at the head of the CQ MUST be returned to the Consumer as a Work Completion. Note that the resources used are expected to be directly accessible by a Non-Privileged Mode Consumer.

The RI MUST support this Verb and MUST support all of the Input & Output Modifiers, except where noted. For more information, see Section 8.2.4 - Returning Completed Work Requests.

Input Modifiers:

* RNIC Handle
* CQ Handle.

Output Modifiers:

* The Work Completion. If an entry is present on the CQ and if the operation completed successfully, this contains information relating to a completed Work Request. If the status of the operation that generates the Work Completion is anything other than success, the contents of the Work Completion are undefined except as noted below. The contents of a Work Completion are:
  - o The 64-bit Work Request ID set by the Consumer in the associated Work Request. This is always valid, regardless of the status of the operation.
  - o The operation type specified in the completed Work Request. The valid operation types are:
    - + Send (for WRs posted to the Send Queue)
    - + Send with Solicited Event (for WRs posted to the Send Queue)

        + Send with Invalidate (for WRs posted to the Send Queue)

        + Send with Solicited Event & Invalidate (for WRs posted to the Send Queue)

        + RDMA Write (for WRs posted to the Send Queue)

        + RDMA Read (for WRs posted to the Send Queue)

        + RDMA Read with Invalidate Local STag (for WRs posted to the Send Queue)

        + Memory Window Bind (for WRs posted to the Send Queue)

        + Fast-Register Non-Shared Memory Region (for WRs posted to the Send Queue)

        + Invalidate Local STag (for WRs posted to the Send Queue)

        + Receive (for WRs posted to the Receive Queue)

    o The number of bytes transferred. This is only valid if the operation type was a Receive.

    o The Completion Status of the operation. This modifier MUST be as specified in Section 9.5.2 - Completion Status Codes.

    o STag Invalidated Indicator. This indicates that the incoming Untagged Message destined for the RQ was a Send with Invalidate or Send with Solicited Event & Invalidate, and thus the STag Invalidated field is valid.

    o STag Invalidated. This contains the STag which was Invalidated. This is only valid when the Invalidated STag Indicator is set.

    o QP ID. This is the QP ID of the QP where the WR which generated this completion was posted.

* Verb Results:

    o Operation completed successfully.

    o Invalid RNIC handle.

    o Invalid CQ handle.

    o CQ empty.

9.3.2.2 Request Completion Notification

Description:

Requests the CQ event handler be called when the next CQE of the specified type is added to the specified CQ.

A CQ event handler must be specified prior to calling this routine (see Section 9.4.1 - Set Completion Event Handler). If the CQ event handler has not been registered when the event is generated, the handler will not be called.

Once the handler routine has been invoked, the Consumer must call Request Completion Notification again to be notified when a new entry is added to that CQ.

It is the responsibility of the Consumer to call the Poll for Completion Verb to retrieve a Work Completion after the handler is called.

The RI MUST support this Verb and MUST support all of the Input & Output Modifiers, except where noted. For more information, see Section 8.2.5 - Asynchronous Completion Notification.

Input Modifiers:

* RNIC Handle.
* CQ Handle.
* Completion notification type. This MUST be either the next completion event or the next solicited completion event.

Output Modifiers:

* Verb Results:
    - Operation completed successfully.
    - Invalid RNIC handle.
    - Invalid CQ handle.

9.4 Event Handling

9.4.1 Set Completion Event Handler

Description:

A RNIC MUST support one CQ Event Handler, and MAY support additional Completion Event Handlers. Each Completion Event Handler address is maintained by the RI and delineated by an opaque handle called a Completion Event Handler Identifier. The consumer uses the Set Completion Event Handler to register individual Completion Event Handlers and obtain a unique Completion Event Handler Identifier. The Completion Event Handler Identifier is used in Create CQ to associate a CQ with a specific Completion Event Handler.

This call does not automatically request a notification on a completion event. The Request Completion Notification Verb must be called in order to request notification.

The RI MUST support this Verb and MUST support all of the Input & Output Modifiers, except where noted. For more information, see Section 8.2.5 - Asynchronous Completion Notification.

Input Modifiers:

* RNIC Handle

* Completion Event Handler Address. If set to zero, then the Set Completion Handler Verb is being used to clear the associated Completion Event Handler address identified by the Completion Event Handler Identifier. The Completion Event Handler will be invoked when an appropriate Completion occurs with the following input parameters passed in to it:

    o RNIC Handle.

    o CQ Handle.

* Completion Event Handler Identifier – An opaque handle used to identify a Completion Event Handler address.

    o If set to zero, the Set Completion Event Handler verb is being used to register a new Completion Event Handler address and the verb will return a new Completion Event Handler Identifier.

    o If set to non-zero, then the Set Completion Event Handler is being used:

        + to clear the associated Completion Event Handler address for the specified Completion Event Handler Identifier, if the Completion Event Handler address is zero;

        + to modify the associated Completion Event Handler address for the specified Completion Event Handler

Identifier, if the Completion Event Handler address is non-zero.

Output Modifiers:

* Completion Event Handler Identifier – Only returned if the Set Completion Event Handler verb is being used to register a new Completion Event Handler address.

* Verb Results:

  o Operation completed successfully.

  o Invalid RNIC Handle.

  o Invalid Completion Event Handler Identifier.

  o Insufficient Resources.

9.4.2 Set Asynchronous Event Handler

Description:

Registers the asynchronous event handler. Only one asynchronous event handler can be registered per RNIC. Additional calls to this Verb will overwrite the handler routine to be called. Additional calls will not generate an additional handler routine. If the new handler address is zero, there will be no Asynchronous Event Handler associated with the RNIC.

The RI MUST support this Verb and MUST support all of the Input & Output Modifiers, except where noted. For more information, see Section 8.3.3 - Asynchronous Errors.

Input Modifiers:

* RNIC Handle

* Asynchronous Event Handler Address. This routine will be invoked with the following input parameters passed in:

  o RNIC Handle.

  o Event Record. This contains information which indicates the resource type and identifier as well as which event occurred:

    + Resource Indicator. This indicates the type of resource to which the Resource Identifier refers. This must be one of the following values:

        * QP
        * CQ
        * RNIC
        * S-RQ

    + Resource Identifier. This value is the QP Handle, CQ Handle, S-RQ Handle or RNIC Handle for the Asynchronous Event.

    + Event Identifier. This indicates the event which caused the Asynchronous Event to be generated. The possible list of Event Identifiers can be found in Section 9.5.3 - Asynchronous Event Identifiers.

Output Modifiers:

* Verb Results:

    o Operation completed successfully.

    o Invalid RNIC Handle.

9.5 Result Types

The following section is a summary of Verb results detailed in Sections 9.2 - 9.4)

9.5.1 Immediate Status Codes

Operation completed successfully – The Verb was executed successfully.	All Verbs

9.5.1.1 RNIC Management Verb Status

Insufficient resources to complete request – An error was detected due to insufficient resources.	Open RNIC, Query RNIC
Invalid Modifier – One of the parameters were invalid.	Open RNIC
Block List mode not supported - The RNIC does not support Block List mode and Block List mode was requested.	Open RNIC
RNIC in use - The RNIC was already in use.	Open RNIC
Invalid RNIC handle - An invalid RNIC handle was specified.	Query RNIC, Close RNIC

Figure 28 - RNIC Management Verb Status

9.5.1.2 PD Management Verb Status

Insufficient resources to complete request – An error was detected due to insufficient resources.	Allocate PD
Invalid RNIC handle - An invalid RNIC handle was specified.	Allocate PD, Deallocate PD
Invalid PD ID - An invalid PD was specified.	Deallocate PD
Protection Domain is in use - The PD was currently in use by a QP, Memory Region, or Memory Window.	Deallocate PD

Figure 29 - PD Management Verb Status

9.5.1.3 CQ Management Verb Status

Insufficient resources to complete request – An error was detected due to insufficient resources.	Create CQ, Modify CQ
Number of CQE requested exceeds RNIC capability – Too many CQ entries for this RNIC were requested.	Create CQ, Modify CQ
An Attempt to shrink the size of the queue failed because too many elements were still present.	Modify CQ
Invalid RNIC handle - An invalid RNIC handle was specified.	Create CQ, Query CQ, Modify CQ, Destroy CQ, Poll CQ
Invalid CQ handle- An invalid CQ handle was specified.	Query CQ, Modify CQ, Destroy CQ, Poll CQ
CQ In Use - One or more QPs is still tied to the CQ.	Destroy CQ
CQ empty - There were no Work Completions available to be retrieved.	Poll CQ
Invalid Completion Event Handler Identifier – An invalid identifier was specified.	Create CQ

Figure 30 - CQ Management Verb Status

9.5.1.4 S-RQ Management Verb Status

Insufficient resources to complete request – An error was detected due to insufficient resources.	Create S-RQ, Modify S-RQ
Invalid RNIC handle - An invalid RNIC handle was specified.	Create S-RQ, Query S-RQ, Modify S-RQ, Destroy S-RQ
Invalid PD ID - An invalid PD was specified.	Create S-RQ
Maximum number of Work Requests requested exceeds RNIC capability.	Create S-RQ, Modify S-RQ

Maximum number of scatter/gather elements per Receive Queue Work Request requested exceeds RNIC capability.	Create S-RQ
S-RQ Limit out of range	Create S-RQ, Modify S-RQ
Invalid S-RQ handle	Query S-RQ, Modify S-RQ, Modify S-RQ
An attempt to shrink the size of the queue failed because too many elements were still present	Modify S-RQ
QPs still associated with the S-RQ	Modify S-RQ
Invalid Input Modifer	Modify S-RQ

Figure 31 - S-RQ Management Verb Status

9.5.1.5 QP Management Verb Status

Insufficient resources to complete request – An error was detected due to insufficient resources.	Create QP, Modify QP
Invalid RNIC handle - An invalid RNIC handle was specified.	Create QP, Query QP, Modify QP, Destroy QP
Invalid CQ handle – An invalid CQ handle was specified.	Create QP
Value requested for ORD exceeds RNIC capability.	Create QP, Modify QP
Value requested for IRD exceeds RNIC capability.	Create QP, Modify QP
Maximum number of Work Requests requested exceeds RNIC capability.	Create QP, Modify QP
Maximum number of scatter/gather elements requested per Work Request exceeds RNIC capability.	Create QP, Modify QP
Invalid PD ID - The PD ID provided was not valid	Create QP
Invalid QP ID – An invalid QP handle was specified.	Query QP, Modify QP, Destroy QP

Cannot change QP attribute - An attempt was made to modify an attribute which is not allowed by the RNIC (for example, number of WQEs)	Modify QP
An Attempt to shrink the size of the queue failed because too many elements were still present.	Modify QP
Invalid state – An invalid QP state was specified.	Modify QP
Invalid LLP Stream handle	Modify QP
Invalid Modifier – One of the modifiers was invalid or was not allowed to be modified in the current state or state transition.	Modify QP
RI Still flushing WQEs - The QP is in the Error state and a request to transition to the Idle state but the RI is still flushing WQEs and therefore cannot transition.	Modify QP
Invalid S-RQ handle	Create QP
QP RQ Limit Out of Range.	Create QP, Modify QP
Memory Windows still Bound to QP	Destroy QP

Figure 32 - QP Management Verb Status

9.5.1.6 Memory Management Verb Status

Insufficient resources to complete request – An error was detected due to insufficient resources.	Allocate NS MR STag, RI-Register, RI-Reregister, Register Shared MR, Allocate MW
Invalid RNIC handle - An invalid RNIC handle was specified.	Allocate NS MR STag, RI-Register,Query MR, Deallocate STag, RI-Reregister, Register Shared MR, Allocate MW, Query MW
Invalid PD ID - An invalid PD ID was specified.	Allocate NS MR STag, RI-Register, RI-Reregister, Register Shared MR, Allocate MW
Invalid Virtual Address - An invalid Memory Address or Offset was specified.	RI-Register, RI-Reregister, Register Shared MR

Invalid Length - An invalid Length was specified. Too many pages or the MR length was too long.	RI-Register, RI-Reregister
Invalid Access Rights requested - An invalid Access Control specifier was specified.	RI-Register, RI-Reregister, Register Shared MR
Invalid Physical Buffer List entry.	RI-Register, RI-Reregister
Invalid Physical Buffer size - The Physical Buffer size (Page/Block)_requested was not supported by the RNIC.	RI-Register, RI-Reregister
Invalid STag Index - An invalid Memory Region STag Index was specified.	Query MR, RI-Reregister, Deallocate STag, Register Shared MR, Query MW
Invalid FBO - the FBO is larger than the physical buffer size	RI-register, RI-Reregister
One or more Memory Windows is still Bound to the Region.	Deallocate STag, RI-Reregister,

Figure 33 - Memory Management Verb Status

9.5.1.7 Post Verb Status

Invalid RNIC handle - An invalid RNIC handle was specified.	PostSQ, PostRQ
Invalid QP handle – An invalid QP handle was specified.	PostSQ, PostRQ
Invalid S-RQ handle - An invalid S-RQ handle was specified.	PostRQ
Too many Work Requests posted.	PostSQ, PostRQ
Invalid Operation type	PostSQ
Invalid QP state.	PostSQ, PostRQ
Invalid Scatter/Gather list format	PostSQ, PostRQ
Invalid Scatter/Gather list length - The Work	PostSQ, PostRQ

Request specified more Scatter/Gather elements than the QP can support.	
RQ Associated with S-RQ - This QP is associated with an S-RQ and therefore the QP Handle cannot be used to post receive Work Requests. The S-RQ handle should be used instead.	PostRQ
Invalid Modifier – One of the parameters were invalid.	PostSQ, PostRQ

Figure 34 - Post Verb Status

9.5.1.8 Event Management Verb Status

Invalid RNIC handle - An invalid RNIC handle was specified.	Request Completion Notification, Set Completion Event Handler, Set Asynchronous Event Handler
Invalid CQ handle - An invalid CQ handle was specified.	Request Completion Notification
Invalid Notify Type – An invalid CQ Notification type was specified.	Request Completion Notification
Invalid Completion event handler identifier – An invalid identifier was specified while attempting to clear a Completion Event Handler address.	Set Completion Event Handler
Insufficient Resources - The RI did not have sufficient resources to complete the request, such as when the Consumer requests another Completion Event Handler Identifier but has already set an amount equal to the value returned in Query RNIC.	Set Completion Event Handler

Figure 35 - Event Management Verb Status

9.5.2 Completion Status Codes

Success – The RNIC Operation was successful.	Send Operation Types, Receive, RDMA Write, RDMA Read, RDMA Read with Invalidate Local STag, Bind, Fast-Register, Invalidate Local STag
Flushed - The Work Request was incomplete when the QP entered the Error state.	Send Operation Types, Receive, RDMA Write, RDMA Read, RDMA Read with Invalidate Local STag, Bind, Fast-Register, Invalidate Local STag
Invalid WQE - The Work Request Element contained a format error.	Send Operation Types, Receive, RDMA Write, RDMA Read, RDMA Read with Invalidate Local STag, Bind, Fast-Register, Invalidate Local STag
Local QP Catastrophic Error - An error related to the QP occurred while processing the Work Request.	Send Operation Types, Receive, RDMA Write, RDMA Read, RDMA Read with Invalidate Local STag, Bind, Fast-Register, Invalidate Local STag
Remote Termination Error - A Terminate Message was received from the Remote Peer that appears to be related to the execution of this Work Request. The error type can be examined by looking at the Terminate Message buffer via Query QP.	Send Operation Types, RDMA Write, RDMA Read, RDMA Read with Invalidate Local STag
Invalid STag - An invalid STag was found in the local SGL. The STag was either not found allocated, bound, or registered in the RI, or an STag of zero was specified for a QP without Privileged rights, or referred to a Shared Memory Region, or the type of STag supplied was not allowed to be used in the specified operation.	Send Operation Types, Receive, RDMA Write, RDMA Read, RDMA Read with Invalidate Local STag, Bind, Fast-Register, Invalidate Local STag

Base & Bounds Violation - The local SGL referenced an address beyond the limits specified for the MR or MW. This includes length errors. For a Bind, the MW was not wholly contained in the MR.	Send Operation Types, Receive, RDMA Write, RDMA Read, RDMA Read with Invalidate Local STag, Bind
Access Violation - The RNIC attempted to read or write to a local SGL MR or MW that did not provide appropriate Access Rights. For a Bind, the MW Access Rights were not compatible with the MR Access Rights.	Send Operation Types, Receive, RDMA Write, RDMA Read, RDMA Read with Invalidate Local STag, Bind
Invalid PD ID - For one of the STags specified in the Work Request the PD of the MR STag was not the same as the PD of the QP, or, the QP of the MW STag was not the same as QP.	Send Operation Types, Receive, RDMA Write, RDMA Read, RDMA Read with Invalidate Local STag, Bind, Fast-Register, Invalidate Local STag
Wrap Error - The specified Address or offset (TO or MO) added to the length of the operation resulted in a wrap beyond the machine-supported address.	Send Operation Types, Receive, RDMA Write, RDMA Read, RDMA Read with Invalidate Local STag, Bind, Fast-Register
STag to Invalidate had Invalid PD or Access Rights - The Invalidate STag on a Receive did not have a PD ID that matched the PD ID of the QP (for a MR) or a QP ID that matched the QP ID of the QP (for a MW). Or the STag did not have Access Rights to be invalidated remotely.	Receive
Zero RDMA Read Resources - The QP ORD value was set to zero.	RDMA Read, RDMA Read with Invalidate Local STag
QP Not In Privileged Mode - The QP is not enabled to perform the Privileged WR.	Fast-Register
STag Not In Invalid state - The STag was already registered or bound, when attempting to Register or Bind it.	Bind, Fast-Register
Invalid Page Size - The page size requested was not supported by the RNIC.	Fast-Register
Invalid Physical Buffer Size - size not supported by the RNIC.	Fast-Register

Invalid Physical Buffer List entry - for page mode, the entry must start on page size boundaries.	Fast-Register
Invalid FBO - the FBO is larger than the physical buffer size.	Fast-Register
Invalid length - requested length is larger than supported by the buffer list.	Fast-Register
Invalid Access Rights specified.	Fast-Register
Physical Buffer List too long.	Fast-Register
Invalid Virtual Address - VA and FBO are not consistent.	Fast-Register
Invalid Region - The STag specified for the MR in the BIND request was invalid.	Bind
Invalid Window - The STag specified for the MW in the BIND request was invalid.	Bind
Invalid Length - The total size of the data to be moved as specified by the sum of the SGL elements, was larger than that supported by the RNIC.	Send, Receive, RDMA Write, RDMA Read, RDMA Read with Invalidate Local STag

Figure 36 - Completion Status Codes

9.5.3 Asynchronous Event Identifiers

The following table contains the list of Event Identifiers and Resource Indicators that the RNIC MUST support as Asynchronous Event Identifiers to be returned by the Asynchronous Event Handler. Note that the Resource Indicator dictates that the appropriate Resource Identifier corresponding to that Resource Indicator MUST be returned as well. For more information, see Section 9.4.2 - Set Asynchronous Event Handler.

Event Identifier and Description.	Resource Indicator
LLP Close Complete – The RDMA Stream has completed Closing and no SQ WQEs were flushed.	QP ID
Terminate Message Received	QP ID
LLP Connection Reset - An incoming LLP Reset (e.g. RST on TCP) was received.	QP ID
LLP Connection Lost	QP ID
LLP Integrity Error: Segment size invalid	QP ID
LLP Integrity Error: Invalid CRC	QP ID
LLP Integrity Error: Bad FPDU - Received MPA marker and 'Length' fields do not agree on the start of a FPDU	QP ID
Remote Operation Error: Invalid DDP version - caused by an inbound segment.	QP ID
Remote Operation Error: Invalid RDMA version - caused by an inbound segment.	QP ID
Remote Operation Error: Unexpected Opcode - caused by an inbound segment.	QP ID
Remote Operation Error: Invalid DDP Queue Number - caused by an inbound segment.	QP ID
Remote Operation Error: Invalid RDMA Read Request Message, RDMA Read not enabled - caused by an inbound segment.	QP ID
Remote Operation Error: Invalid RDMA Write or RDMA Read Response Message, RDMA Write & RDMA Read Response not enabled - caused by an inbound segment.	QP ID
Remote Operation Error: Invalid RDMA Read Request Message, message size too small or Offset non-zero - caused by an inbound segment.	QP ID
Remote Operation Error: No 'L' bit when expected - caused by an inbound segment.	QP ID

Protection Error: Invalid STag - caused by an inbound Tagged DDP segment not valid for this QP. This includes using the STag of zero, the STag was not associated with the QP or the STag was in the Invalid state.	QP ID
Protection Error: Tagged Base and bounds violation - caused by an inbound Tagged segment attempted to access memory outside the limits assigned to the STag.	QP ID
Protection Error: Tagged Access Rights violation - caused by an inbound segment referencing a Tagged Buffer which did not have the necessary memory Access Rights for the requested operation.	QP ID
Protection Error: Tagged Invalid PD - caused by an inbound segment referencing a Tagged Buffer which was not allowed to be referenced by QP.	QP ID
Protection Error: Wrap error - caused by an inbound segment not targeting the RQ.	QP ID
Bad Close - The QP was in the Closing state when a Segment arrived.	QP ID
Bad LLP Close - An attempt was made to close the RDMA Stream with work in progress.	QP ID
RQ Protection Error – Invalid MSN – MSN range not valid. Caused by an inbound segment targeting the RQ. Possibly due to Receive Queue being empty.	QP ID
RQ Protection Error – Invalid MSN – gap in MSN. Caused by an inbound segment targeting the RQ.	QP ID
IRRQ Protection Error: Invalid MSN - too many RDMA Read Request Messages in progress - caused by an inbound segment not targeting the IRRQ.	QP ID
IRRQ Protection Error: Invalid MSN - gap in MSN - caused by an inbound segment not targeting the RQ.	QP ID
IRRQ Protection Error: Invalid MSN - range is not valid - caused by an inbound segment not targeting the RQ.	QP ID
IRRQ Protection Error: Invalid STag - Data Source STag determined to be invalid during RDMA Read Response	QP ID

processing.	
IRRQ Protection Error: Tagged Base and bounds violation - This includes RDMA Read Request of a message larger than supported by the RNIC. It is detected accessing the Data Source during RDMA Read Response processing.	QP ID
IRRQ Protection Error: Tagged Access Rights violation - Data Source Access Rights violation detected during RDMA Read Response processing.	QP ID
IRRQ Protection Error: Tagged Invalid PD - Data Source PD violation detected during RDMA Read Response processing.	QP ID
IRRQ Protection Error: Wrap error - detected during RDMA Read Response processing.	QP ID
CQ/SQ Error: CQ Overflow Error - An error occurred on the CQ during a SQ completion.	QP ID
CQ/RQ Error: CQ Operation error - An error occurred on the CQ during a RQ completion.	QP ID
S-RQ error on a QP – An error occurred while attempting to pull a WQE from the S-RQ associated with the QP.	QP ID
Local QP Catastrophic Error – occurred during processing.	QP ID
CQ Overflow Detected - An overflow of the Completion Queue has been detected. This Error Code is OPTIONAL.	CQ Handle
CQ Operation Error– An error occurred on the CQ unrelated to a specific QP completion.	CQ Handle
Shared Receive Queue Limit reached - The Limit value established for the Shared Receive Queue has been reached.	S-RQ Handle
QP RQ Limit Reached - The Limit value established for the QP's RQ has been reached.	QP ID
Shared Receive Queue Catastrophic Failure - A problem occurred with the RNIC or its driver that renders the RNIC unable to use the S-RQ.	S-RQ Handle

RNIC Catastrophic Failure - A problem occurred with the RNIC or its driver that renders the RNIC unable to reliably function.	RNIC Handle

Figure 37 - Asynchronous Event Identifiers

10 Security Considerations

Security Considerations are necessary for the RDMA Protocols and this specification. An Internet Draft is under development.

11 IANA Considerations

If DDP was enabled a priori for a ULP by connecting to a well-known port, this well-known port would be registered for the DDP with IANA.

12 References

12.1 Normative References

[RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[MPA] P. Culley et al., "Markers with PDU Alignment", RDMA Consortium Draft Specification draft-cully-iwarp-mpa-00.doc, October 2002

[DDP] H. Shah et al., "Direct Data Placement over Reliable Transports", RDMA Consortium Draft Specification draft-shah-iwarp-ddp-00.txt, October 2002

[RDMAP] R. Recio et al., "RDMA Protocol Specification", RDMA Consortium Draft Specification draft-recio-iwarp-00, October 2002

[SCTP] R. Stewart et al., "Stream Control Transmission Protocol", RFC 2960, October 2000.

[TCP] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

12.2 Informative References

[IPSEC] Atkinson, R., Kent, S., "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

13 Appendix

13.1 Connection Initialization at LLP Startup

The purpose of an initialization at LLP Startup is to enable iWARP using the minimum number of messages possible. Note that not all RNIC/OS implementations are required to support this.

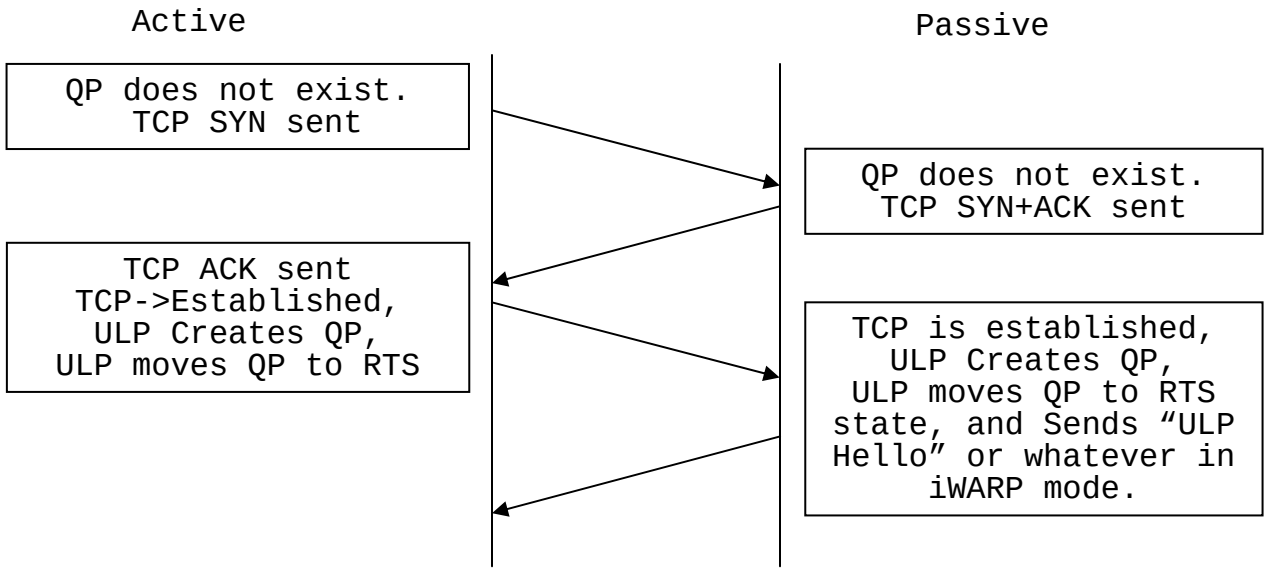


Figure 39 - Connection Initialization at LLP Startup (using TCP)

Below is an example sequence for an iWARP startup that accomplishes this (other sequences are possible). The Sequence applies equally to either the active or passive side.

* The Consumer establishes the LLP Connection using a non-Verbs interface.

* The Consumer creates a QP, setting up the CQ, PD, etc., and registers memory for buffers.

* The Consumer posts buffers to the RQ appropriate for the expected traffic.

* If the ULP intends to transmit first, the Consumer could Post one or more Work Request(s) on the SQ (usually a SEND message) that will be sent after the QP is placed in the RTS state.

* The Consumer moves the QP state to RTS. The Modify QP Verb for this includes the LLP Stream Handle, and does not include a streaming message buffer.

* If the local Consumer intends to perform RDMA Read Type WRs, the local Consumer obtains, in some ULP defined message, the number of incoming RDMA Read Request Messages that the Remote Peer can have outstanding (IRD). If the Remote Peer's IRD is smaller than the local Peer's ORD, the local Consumer should also perform a Modify QP Verb with the Remote Peer's IRD value placed into the local ORD value prior to posting the first RDMA Read Type WR. The local Consumer may also transmit, in some ULP defined message, the number of outgoing RDMA Read Request Messages that the Local Peer can have outstanding (ORD).

* If the local Consumer intends the QP to be the Data Source of RDMA Read Operations, the Consumer provides, in some ULP defined message, the number of incoming RDMA Read Request Messages (e.g. IRRQ depth) that the Local Peer can have outstanding (IRD). The Consumer may also receive, in some ULP defined message, the number of outgoing RDMA Read Request Messages that the Remote Peer can have outstanding (ORD). If the Remote Peer's ORD is smaller than the Local Peer's IRD, the local Consumer may also perform a Modify QP Verb with the Remote Peer's ORD value placed into the local IRD value prior to posting the first RDMA Read Type WR.

This specification does not define which side of the connection sends the first message, the active or passive side; the ULP is responsible for determining this. In addition, this specification does not preclude the use of Active/Active connections.

RNIC Implementers note: Since there is no integration between the RI and the LLP Connection startup sequence, as defined above, it is possible that some data may arrive over the transport before the RNIC is in iWARP mode. It is the responsibility of the RI to accept this data and interpret it as iWARP data. Alternately, the Consumer (or other service that establishes the LLP Connection) can ensure that no data will be received prior to moving the QP to RTS state. If neither of these methods is available, then iWARP startup with the LLP is not available.

13.2 Graceful Receive Overflow Handling

A valid implementation option is to gracefully handle Receive Queue or Shared-Receive Queue overflow. In a strictly layered model, this may be difficult but in an RNIC implementation, this should be feasible.

In the current architecture, if there are no Receive Queue Work Queue Elements available when an Untagged Message arrives then the connection is dropped. This is true if there is a Shared Receive Queue or a dedicated receive queue.

In this case, the implementation (RI/RNIC), which is not relying on an external LLP, may choose to handle this gracefully through LLP mechanisms. In this case, the RI will choose to not drop the connection and instead appear to pause receive queue processing until more WQEs have been posted to the RQ or S-RQ.

How the RNIC decides to perform this function is left up to implementation. One example mechanism which may be used to gracefully handle receive overflow is for the implementation to drop incoming packets when there are no WQEs on the RQ or S-RQ. This type of mechanism may have side effects, such as causing back-off algorithms to be invoked, but this type of mechanism is still a valid implementation option.

14 Author's Addresses

Jeff Hilland
Hewlett-Packard Company
20555 SH 249
Houston, TX 77070-2698 USA
Phone: +1 (281) 514-9489
Email: jeff.hilland@hp.com

Paul R. Culley
Hewlett-Packard Company
20555 SH 249
Houston, TX 77070-2698 USA
Phone: +1 (281) 514-5543
Email: paul.culley@hp.com

James Pinkerton
Microsoft Corporation
One Microsoft Way
Redmond, WA. 98052 USA
Phone: +1 (425) 705-5442
Email: jpink@windows.microsoft.com

Renato Recio
IBM Corporation
11501 Burnett Road
Austin, TX 78758 USA
Phone: +1 (512) 838-1365
Email: recio@us.ibm.com

15 Acknowledgments

John Carrier
Adaptec, Inc.
691 S. Milpitas Blvd.
Milpitas, CA 95035 USA
Phone: +1 (360) 378-8526
Email: john_carrier@adaptec.com

Hari Ghadia
Adaptec, Inc.
691 S. Milpitas Blvd.,
Milpitas, CA 95035 USA
Phone: +1 (408) 957-5608
Email: hari_ghadia@adaptec.com

Patricia Thaler
Agilent Technologies, Inc.
1101 Creekside Ridge Drive, #100
M/S-RG10
Roseville, CA 95678
Phone: +1 (916) 788-5662
email: pat_thaler@agilent.com

Mike Penna
Broadcom Corporation
16215 Alton Parkway
Irvine, California 92619-7013 USA
Phone: +1 (949) 926-7149
Email: MPenna@Broadcom.com

Uri Elzur
Broadcom Corporation
16215 Alton Parkway
Irvine, California 92619-7013 USA
Phone: +1 (949) 585-6432
Email: Uri@Broadcom.com

Ted Compton
EMC Corporation
Research Triangle Park, NC 27709, USA
Phone: +1 (919) 248-6075
Email: compton_ted@emc.com

Dwight Barron
Hewlett-Packard Company
20555 SH 249
Houston, TX 77070-2698 USA
Phone: +1 (281) 514-2769
Email: Dwight.Barron@Hp.com

Mallikarjun Chadalapaka
Hewlett-Packard Company
8000 Foothills Blvd.
Roseville, CA 95747-5668, USA
Phone: +1 (916) 785-5621
Email: cbm@rose.hp.com

Dave Garcia
Hewlett-Packard Company
19333 Vallco Parkway
Cupertino, Ca. 95014 USA
Phone: +1 (408) 285-6116
Email: dave.garcia@hp.com

Mike Krause
Hewlett-Packard Company, 43LN
19410 Homestead Road
Cupertino, CA 95014 USA
Phone: +1 (408) 447-3191
Email: krause@cup.hp.com

Jim Wendt
Hewlett-Packard Company
8000 Foothills Boulevard
Roseville, CA 95747-5668 USA
Phone: +1 (916) 785-5198
Email: jim_wendt@hp.com

John L. Hufferd
IBM Corp.
650 Harry Rd.
San Jose CA
Phone: +1 (408) 256-0403
Email: hufferd@us.ibm.com

Mike Ko
IBM Corp.
650 Harry Rd.
San Jose, CA 95120, USA
Phone: +1 (408) 927-2085
Email: mako@us.ibm.com

Ellen Deleganes
Intel Corporation
MS JF5-355
2111 NE 25th Ave.
Hillsboro, OR 97124 USA
Phone: +1 (503) 712-4173
Email: ellen.m.deleganes@intel.com

Frank Berry
Intel Corporation
2111 NE 25th Ave.
Hillsboro, OR 97124 USA
Phone: +1 (503) 712-3897
Email: frank.berry@intel.com

Howard C. Herbert
Intel Corporation
MS CH7-404
5000 West Chandler Blvd.
Chandler, AZ 85226 USA
Phone: +1 (480) 554-3116
Email: howard.c.herbert@intel.com

Dave Minturn
Intel Corporation
MS JF1-210
5200 North East Elam Young Parkway
Hillsboro, OR 97124 USA
Phone: +1 (503) 712-4106
Email: dave.b.minturn@intel.com

Hemal Shah
Intel Corporation
MS PTL1
1501 South Mopac Expressway, #400
Austin, TX 78746 USA
Phone: +1 (512) 732-3963
Email: hemal.shah@intel.com

James Livingston
NEC Solutions (America), Inc.
7525 166th Ave. N.E., Suite D210
Redmond, WA 98052-7811
Phone: +1 (425) 897-2033
Email: james.livingston@necsam.com

Tom Talpey
Network Appliance
375 Totten Pond Road
Waltham, MA 02451 USA
Phone: +1 (781) 768-5329
Email: thomas.talpey@netapp.com

16 Full Copyright Statement