

Overview

The purpose of this document is to describe a mapping of SCSI commands between Fibre Channel (FC) and InfiniBand (IB) transports. It also provides a brief reference for the FC, IB, SCSI, FCP, and SRP standards. Whereas FCP and SRP specify mappings of SCSI commands onto each of these transports, this proposal is intended to provide a flexible method to convert between them, referred to here as bridging.

The first section describes the bridging proposal.

The second section describes the relevant aspects of the Fibre Channel standard as specified in (see www.t11.org):

- *Fibre Channel – Switch Fabric - 2 (FC-SW-2)*, Revision 5.4, 2001-Jun-26
- *Fibre Channel - Framing and Signaling (FC-FS)*, Version 1.20, 2001-Feb-16
- *Fibre Channel – Fabric Generic Requirements (FC-FG)*, Revision 3.5, 1996-Aug-7.

The third section describes the relevant aspects of the InfiniBand Architecture as specified in (see www.infinibandta.org):

- *InfiniBand™ Architecture Specification Volume 1* Release 1.0.a, dated 2001-Jun-19
- *InfiniBand™ Architecture Specification Volume 3* Release 0.9 Draft, dated 2000-Sep-29
- *InfiniBand™ Architecture Development and Deployment*, by William Futral, dated 2001.

The fourth section describes the relevant aspects of the SCSI architecture as specified in (see www.t101.org):

- *SCSI Architecture Model - 2 (SAM-2)*, Revision 18, dated 2001-May-31
- *SCSI Primary Commands – 3 (SPC-3)*, Revision 01, dated 2001-Sep-22
- *Fibre Channel Protocol for SCSI, Second Version (FCP-2)*, Revision 7, dated 2001-Mar-7
- *SCSI RDMA Protocol (SRP)*, Revision 10, dated 2001-Oct-3
- *IEEE Tutorial for SCSI use of IEEE company_id*, dated 1997-Feb-25
- *Access Controls for SPC-3*, dated 2001-Sep-22,
<ftp://ftp.t10.org/t10/document.01/01-268r2.pdf>.

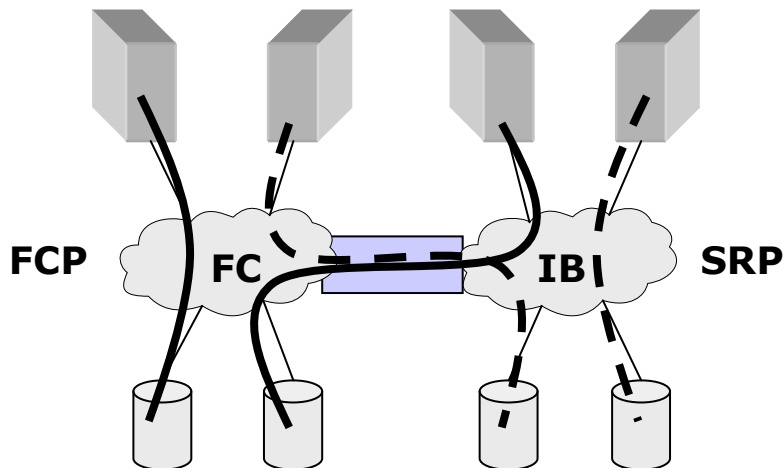
Bridging

The SCSI application layer uses services provided by the SCSI protocol layer to transport transactions between application clients and logical units. When the protocol layer uses multiple physical interconnects to transport these transactions, bridging is required to convert between information units (IU) used by different protocols. The bridging function is transparent to the SCSI device driver acting as the application client in the initiator device and to the logical unit device server within the target device because their transactions occur at the SCSI application layer which use the transport services provided by FCP and SRP. The physical interconnect services used by FCP and SRP may be provided by Fibre Channel, InfiniBand or Virtual Interface. Note that VI is a session layer protocol providing

semantics similar to the IB transport layer, which may be implemented on top of protocols including ATM, FC, and TCP/IP.

Processes communicate using transactions that are segmented into packets suitable for routing over a network. Tunneling may be used to encapsulate packets that flow over a different interconnect than that used by the communicating processes; control information is added to each packet before it is sent over the unlike interconnect and then removed before returning it to its native interconnect. When the communicating processes use different transports, bridging acts as a proxy between the different name spaces, replacing the control information suitable for sending the transaction over one transport with that suitable for another transport. Unlike tunneling which can simply add and remove wire protocol information to each packet, the bridging function must convert transport protocol information and establish the appropriate communication endpoints for packet routing.

The following figure illustrates the environment discussed in this bridging proposal. The device interconnecting the FC fabric and the IB subnet, containing the bridging function, is called a router. The predominant storage networking environment uses FCP to access FC storage from FC hosts. As InfiniBand hosts are deployed they will need to access FC storage using a multiprotocol router. As native InfiniBand storage becomes available (dashed lines) a multiprotocol router will also be used to allow access by FC hosts. InfiniBand hosts use SRP to access native InfiniBand storage. Hosts that are attached to both FC and InfiniBand fabrics would likely use the fabric where the storage is attached rather than accessing the storage through the router though the router can provide redundant paths using both fabrics.



SCSI commands are sent using transport protocol IUs. Communication between SCSI devices over the FC fabric uses FCP. Communication between SCSI devices over the InfiniBand subnet uses SRP. Communication from the FC fabric to the InfiniBand subnet uses the bridging function to map FCP to SRP. Communication from the InfiniBand subnet to the FC fabric uses the bridging function to map SRP to FCP.

Fibre Channel Standard

The Fibre Channel (FC) standard provides a general transport vehicle for Upper Level Protocols (ULPs). It was designed to meet the requirements of reliably transporting large amounts of data between master host systems and slave peripheral devices (channel communication) and of transporting many small messages between peer hosts in unpredictable environments (network communication). Logically, FC is a bi-directional, point-to-point, serial data channel. Physically, FC is an interconnection of multiple N_Ports, interconnected by a switching network, called a Fabric, or a point-to-point link.

FC is structured as a set of hierarchical functions but does not restrict implementations to specific interfaces between these levels:

- FC-4 defines the mapping, between the lower levels of the Fibre Channel and ULPs including the SCSI command set. FC-4 data transferred as a single Sequence by FC-2 is called an Information Unit (IU).
- FC-3 provides a set of services that are common across multiple N_Ports of a FC node.
- FC-2 specifies the rules, and provides mechanisms needed to transfer blocks of data end to end and defines capabilities for use by FC-4. Transported data is transparent to FC-2 and visible to FC-3 and above.
- FC-1 defines the transmission protocol that includes the serial encoding, decoding, and error control.
- FC-0 consists of transmission media, transmitters, and receivers and their interfaces.

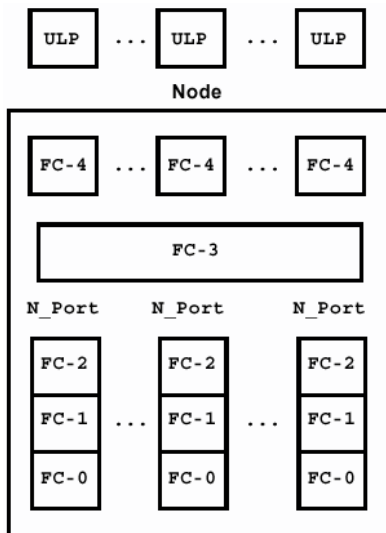
Fibre Channel Port types defined in FC-GS and FC-SW-2 are:

- B_Port is a Fabric inter-element port used to connect Bridge devices with E_Ports on a Switch. The B_Port provides a subset of the E_port functionality.
- E_Port is a Fabric Inter-Element Port used to establish Inter-Element Links (IEL). FC-SW-2 revises the definition to a Fabric "Expansion" Port that attaches to another Interconnect Port to create an Inter-Switch Link. Each E_Port has a unique Name within the Fabric. An E_Port Index is a value associated with an E_Port used by the Fabric Shortest Path First (FSPF) Protocol.
- F_Port is the Link Control Facility within the Fabric which attaches to an N_Port through a link. An F_Port is addressable by the N_Port attached to it, with a common well-known address identifier (hex 'FFFFFF').
- FL_Port is an F_Port that contains Arbitrated Loop functions associated with Arbitrated Loop topology.
- Fx_Port is a Switch Port capable of operating as an F_Port or FL_Port.
- Fabric_Port is a generic reference to an E_Port, F_Port, FL_Port, G_Port, or GL_Port.
- G_Port is a generic Fabric_Port that can function either as an E_Port or as an F_Port.
- GL_Port is a generic Fabric_Port that can function either as an E_Port or as an FL_Port.
- Interconnect_Port is a generic reference to an E_Port or a B_Port.
- L_Port is a generic reference to an FL_Port, a GL_Port, or an NL_Port.
- N_Port is a hardware entity that includes a Link Control Facility. It may act as an Originator, a Responder, or both.
- NL_Port is an N_Port that contains Arbitrated Loop functions associated with Arbitrated Loop topology.

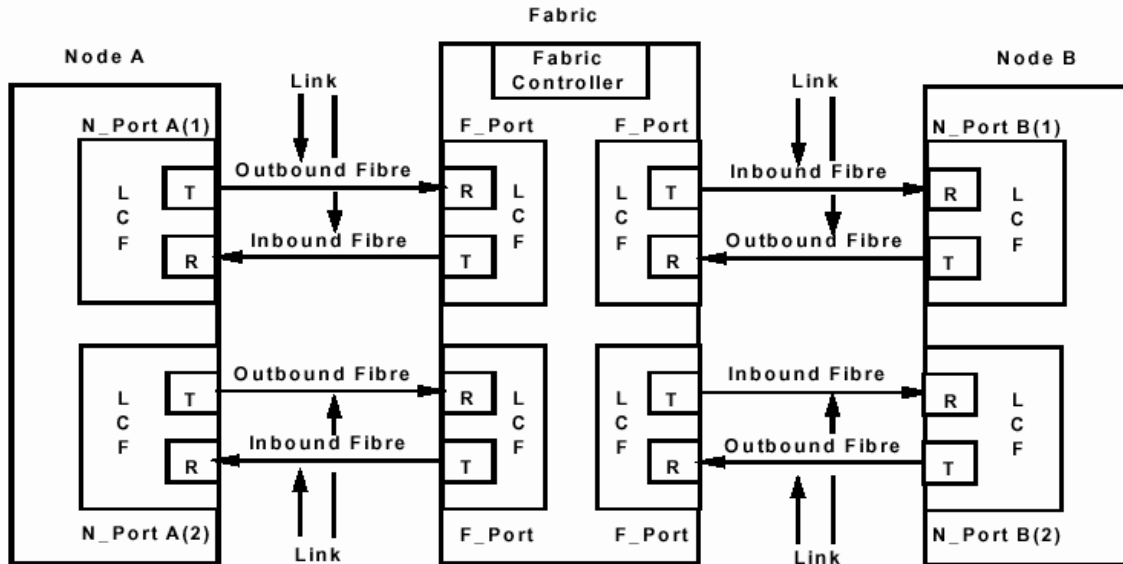
- Nx_Port is a Port operating as an N_Port or NL_Port. N_Ports and NL_Ports capable of supporting FCP transactions are collectively referred to as FCP_Ports by FCP-2.
- Port is a generic reference to an E_Port, F_Port, G_Port, N_Port or S_Port. FC-FS revised the definition to a generic reference to an N_Port or F_Port. FC-SW-2 later revised the definition to a generic reference to an N_Port, NL_Port, F_Port, FL_Port, B_Port, or E_Port.
- S_Port is a Fabric internal service node that functions both as a Fabric_Port and as an N_Port.
- Switch Port is an E_Port, F_Port, or FL_Port.

FC devices support one or more Nodes and each Node contains one or more N_Ports. A Fabric is the entity which interconnects the N_Ports attached to it and is capable of routing frames by using only the D_ID information. A Sub-Fabric is a set of ports and services in a Fabric uniquely identified by one data rate and one Class of service. A Region is a section of a Sub-Fabric with compatible service parameters in which all ports can communicate. An Extended Region is a section of two or more Sub-Fabrics with compatible service parameters forming an extended communication group.

The logical model of a Node is depicted in the following figure.



The physical model of connecting FC nodes through a fabric is depicted in the next figure. A fabric interconnects various N_Ports attached to it and is capable of routing frames by using only the D_ID information. A Link Control Facility (LCF) is a hardware facility that attaches to each end of a link and manages transmission and reception of data. It is contained within each Port and includes a transmitter and a receiver.

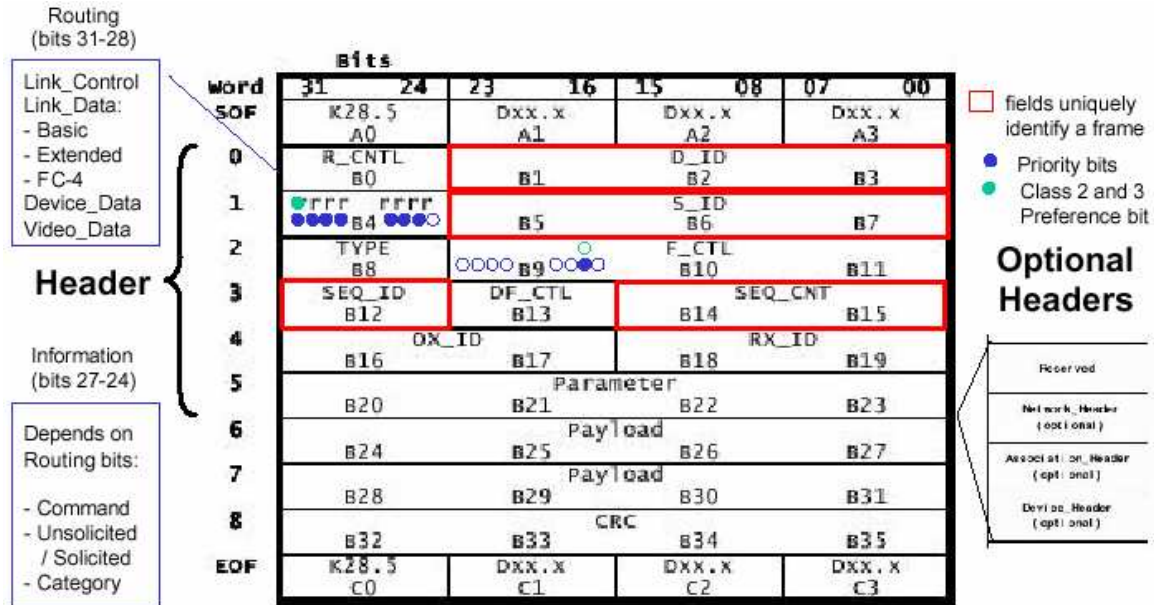


An Exchange is the basic mechanism that transfers one or more related non-concurrent Sequences that may flow in the same or opposite directions between two N_Ports, referred to as the Originator and the Responder. An Exchange is identified by a 2 byte Originator Exchange Identifier (OX_ID) and a 2 byte Responder Exchange Identifier (RX_ID), generically referred to as an Exchange Identifier (X_ID). The OX_ID and RX_ID are assigned by and are only meaningful to the Originator and Responder respectively.

A Sequence is one or more Data frames with a common 1 byte Sequence Identifier (SEQ_ID) transmitted unidirectionally between two N_Ports, referred to as the Initiator and the Recipient. If applicable, the Recipient sends Link Control frames as a response to each Data frame. An N_Port uses a Sequence Qualifier (S_ID, D_ID, OX_ID, RX_ID, and SEQ_ID) to uniquely identify Active and Open Sequences. An Initiator or Recipient is considered Active until all frames in a Sequence have been transmitted or received, respectively. A Sequence (or Exchange) is considered Open during the period of time when it is initiated until it is terminated.

An N_Port transmits Data frames as a result of information transfer requests from its associated FC-4s at its end, and transmits Link Control frames in response to Data frames that it receives. An N_Port may send Data frames, send and receive Data frames simultaneously, or send and receive Data frames serially; in each case, Link Control frames flow in the opposite direction of the Data frames. A frame is uniquely identified by S_ID, D_ID, SEQ_ID, SEQ_CNT, and Sequence Context. The OX_ID and RX_ID fields may be used in place of the S_ID and D_ID to identify the N_Port pair associated with a specific frame. Link Control frames are used to indicate successful or unsuccessful delivery of Data frames, to control the flow of Data frames, and to provide some low-level N_Port commands. The function of a Data frame is categorized as a Device Data frame, a Video Data frame, or a Link Data frame. Link Data frames are used to provide Basic (BLS), Extended (ELS), and FC-4 Link Services, which are requests sent to a destination F_Port or N_Port to perform a function or service. N_Ports must support all BLS commands and the following ELS requests and associated replies: FLOGI, PLOGI, LOGO, RRQ, and ESTC.

A FC frame, consisting of one or more headers, a payload, and a CRC field is depicted in the next figure. The frame header is used to control link operations, control device protocol transfers, and detect missing or out-of-order frames. A frame is transmitted on a word boundary.



A frame is transmitted in the following byte order: A0..A3, B0..B35, C0..C3

The frame header fields related to FC addressing are:

An **Address Identifier (FC_ID)** is an unsigned 24 bit address used to uniquely identify the source (S_ID) and destination (D_ID) of FC frames. The **S_ID** (Word 1, Bits 23-0) and **D_ID** (Word 0, Bits 23-0) Address Identifiers are used to route frames within the fabric. If the address space is partitioned, the Domain Identifier is bits 23-16, the Area Identifier is bits 15-8, and the Port Identifier is bits 7-0. The suggestion is that, for addressing purposes, Ports are members of Areas, Areas are members of domains, and Domains are partitions of the Fabric.

An N_Port Identifier is a fabric unique Address Identifier by which an N_Port is uniquely known. An N_Port may also have one or more Alias Address Identifiers. The F_Port which is directly connected through a Link to an N_Port has the reserved Address Identifier FFFFEEh which is used primarily as the D_ID for Fabric Login.

Where the Fabric provides address assignment, the Fabric first assigns unique address identifiers to each of its F_Ports as part of the initialization procedure. The N_Ports attached to each of the F_Ports then inherit the address identifiers of their associated F_Ports. The F_Port entity is not separately addressable from the N_Ports except that the N_Port can address the F_Port to which it is linked using the reserved F_Port address.

SEQ_ID is a one byte field (Word 3, Bits 31-24) assigned by the Sequence Initiator which shall be unique for a specific D_ID and S_ID pair while the Sequence is Open.

SEQ_CNT is a two-byte field (Word 3, Bits 15-0) that indicates the sequential order of Data frame transmission within a single Sequence or multiple consecutive Sequences for the same Exchange.

OX_ID is a two-byte field (Word 4, Bits 31-16) that shall identify the Exchange ID assigned by the Originator of the Exchange. An OX_ID of hex 'FF FF' indicates that the OX_ID is unassigned and has only one Exchange with a given Responder.

RX_ID is a two byte field (Word 4, Bits 15-0) assigned by the Responder to provide a unique, locally meaningful identifier for an Exchange established by an Originator and identified by an OX_ID. An RX_ID of hex 'FF FF' indicates that the RX_ID is unassigned.

The frame header fields related to FC-4 protocols are:

DF_CTL is a one-byte field (Word 3, Bits 23-16) that specifies the presence of optional headers at the beginning of the Data Field for Device_Data or Video_Data frames.

TYPE is a one-byte field (Word 2, Bits 31-24) that identifies the protocol of the frame content at the Sequence Recipient for Data frames. Some example values are 5h (IP over FC, RFC 2625), 8h (SCSI – FCP), 20h (CT), 23h (FC-AL), 24h (SNMP), and 58h (FC-VI).

The frame header fields related to Quality of Service (QoS) are:

rrrr rrrr is a one byte field (Word 1, bits 31-24) interpreted according to the value of bit 17 of a three byte field **F_CTL** (Word 2, Bits 23-0). When bit 17 is set to one, bits 31-25 are the **Priority** and bit 24 is a preemption request flag. A Priority of x'00' indicates no priority is assigned. The remaining values indicate increasing relative priority (e.g., 1h has a lower priority than 2h). When bit 17 is set to zero, bit 31 indicates whether to deliver a Class 2 or 3 frame with (1) or without (0) **Preference**.

Other FC addressing information:

Name Identifier: A 64 bit identifier, with a 60 bit value preceded with a 4 bit Network Address Authority (NAA) Identifier, used to identify entities in Fibre Channel such as N_Port, Node, F_Port, or Fabric. A Worldwide Name (WWN) is a Name Identifier which is worldwide unique, and represented by a 64 bit unsigned binary value.

FC uses the high order 4 bits (the NAA field) to identify the format of a world-wide unique identifier as IEEE (1h), IEEE extended (2h), IEEE registered (5h), and IEEE registered extended (6h).

The IEEE format is typically used to uniquely identify a FC Node. The high order 4 bits are x'1', followed by 12 bits that are reserved x'000', followed by 12 bits that identify the company, with the low order 12 bits provided by the identified company.

The IEEE Extended format (high order 4 bits are 2h) allows the company to use the reserved 12 bits to extend the company-specified unique value. This format is in

common usage in Fibre Channel devices as a port identifier although some applications also use this format as a node identifier.

The IEEE Registered format specifies the high order 4 bits as 5h, followed by the 24 bit IEEE company ID, followed by 36 bits provided by the identified company. This format should be used to uniquely identify FC Nodes, N_Ports, F_Ports, Fabrics, or other objects except where historical usage makes the previous identifier formats more appropriate.

The IEEE Registered Extended format is defined by the FC-FS, but is used only for SCSI objects attached by Fibre Channel. The high order 4 bits are 6h, followed by the 24 bit IEEE company ID, followed by a 36-bit vendor specified identifier, followed by a 64-bit vendor specified identifier Extension which uniquely identifies the object. The vendor specified identifier extension may be used sparsely or assigned by the identified object as long as uniqueness of the full 128-bit IEEE registered extended identifier is guaranteed. This identifier format is useful in devices like RAID controllers.

The optional header fields related to FC addressing are:

If no optional headers are present, all space in the Data Field may be used for payload. If optional headers are present, the first 16 bytes of the payload are reserved, followed by the Network Header if present, then the Association Header if present, then the Device Header if present.

The 16 byte **Network Header** (present when bit 21 in the DF_CTL field is set to one) is used for routing between different Fabric address spaces or between FC and non-FC networks. It consists of a destination address followed by a source address, each 60 bits prefixed by a 4 bit NAA field identifying the address format. In addition to the IEEE formats identified above, 3h means locally assigned, 4h means IP, 12h means CCITT individual address, 14h means CCITT group address, with other bit combinations are reserved. If used, the Network Header is present only in the first Data frame of a Sequence.

The 32 byte **Association Header** (present when bit 20 in the DF_CTL field is set to one) is used to identify one or more processes within a node, where more than X_ID and SEQ_ID are required for identification. It consists of a 1 byte validity field indicating whether the associators are meaningful and if unicast or multicast is supported, a 7 byte Originator Process Associator, a reserved byte, a 7 byte Responder Process Associator, followed by 16 bytes of zeros. If used, the Association Header is present only in the first Data frame of a Sequence.

The **Device Header** (present when bits 17-16 in the DF_CTL field are not zero) is 16 (01b), 32 (10b), or 64 (11b) bytes in size. The contents of the Device Header are entirely under the control of a level above FC-2 based on the TYPE field.

InfiniBand Architecture

InfiniBand defines a memory-to-memory switched communications fabric for connecting multiple nodes with high bandwidth and low latency in a protected, remotely managed environment. An InfiniBand subnet is composed of end nodes, switches, routers, and subnet managers interconnected by links. Each node may

attach to one or more switches and/or directly with each other. Multiple links can exist between any two nodes. An end node can communicate using multiple ports and multiple paths to provide fault tolerance and increased bandwidth. A port is attached to one subnet at a time. InfiniBand supports a link rate of 2.5 Gbps in each direction with link widths of 1,4,and 12 bits, byte-stripped. InfiniBand transmits the most significant byte of each field first (Big Endian), usually on 32-bit boundaries.

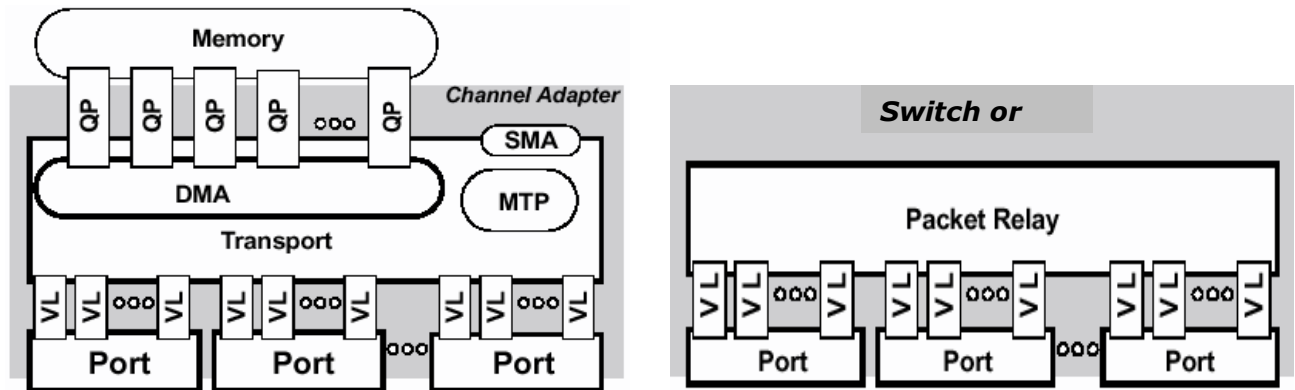
Processes execute transactions consisting of units of work called messages. Hardware segments and re-assembles messages into data packets that are the routable units of transfer. The size of a packet transferred between two end nodes (which may span subnets) is the largest size supported by all ports in the path. In other words, this Path MTU (maximum transmission unit) is the minimum of all port MTUs in the path. The minimum port MTU (which represents the size of the data payload only) is 256 bytes though 512, 1024, 2048, and 4096 byte MTUs are also defined.

Channel Adapters (**CA**) are the InfiniBand devices that generate and consume packets. Host Channel Adapters (HCA) reside in processor nodes while Target Channel Adapters (TCA) reside in I/O units. A CA contains a programmable DMA engine, a Memory Translation and Protection (MTP) mechanism, a Subnet Management Agent (**SMA**), and one or more ports as illustrated in the following figure. The Subnet Manager (**SM**) assigns each port one or more Local Identifiers (**LID**) used for switching within a subnet and one or more Global Identifiers (**GID**) used for routing between subnets. Each CA and CA port also has a globally unique identifier (**GUID**) assigned by the manufacturer.

A switch connects links within a subnet by forwarding packets to an outbound port (unicast) or ports (multicast) based on the packet's Destination LID and Service Level and the switch's forwarding table. A switch has one LID, one GID, and one GUID and is transparent to end nodes. A switch contains up to 255 physical ports and one management logical port 0. Support for partitioning and multicast is optional.

A router connects subnets by forwarding packets based on the packet's destination GID and the router's forwarding table. Each router forwards the packet through the next subnet to another router until the packet reaches the destination subnet. The last router sends the packet using the LID associated with the destination GID. Each router and router port has a GUID. Each router port also has one or more LID and GID. A router is not transparent to end nodes since the source CA must specify the LID of the router and the destination GID.

Quality of Service (QoS) is provided by a value in each packet called a service level (**SL**) that is used to select which virtual lane (**VL**) buffers to use within a port. The SL/VL mappings vary from port to port so packets with different service levels can use the same path. Allowing a packet to use multiple VLs also allows vendors to support QoS while implementing the number of VLs best suited for their device.



The channel interface, described by verbs, provides consumers (the operating system and its applications) access to messaging (send/receive), Remote Direct Memory Access (RDMA-Read, RDMA-Write, Atomic 64-bit Compare & Swap, and Atomic 64-bit Fetch & Add), and memory registration and binding operations over channels between peer work queue pairs. The consumer posts work requests (**WR**) to queue pairs (**QP**) and polls completion queues (**CQ**) to retrieve work completions (**WC**). Each consumer process creates one or more CQs and associates each send and receive queue of a QP to a (potentially different) CQ. A process identifies each QP locally by its QP handle and remotely by its QP number.

RDMA-Read copies data from a virtually continuous remote memory region into local memory according to a scatter list. A reliable RDMA-Read response indicates data sent previously has been received on the remote side. Periodic zero-length RDMA-Reads can be used as a heartbeat since the remote side is alive if it completes. RDMA-Write copies data from local memory regions according to a gather list into a continuous remote memory region. When immediate data (32 bits) is included in an RDMA-Write operation a receive WR must be pre-posted.

A QP is configured for a particular service type that determines how QPs are associated with other QPs and what operations are possible over that channel. A QP configured for Reliable Connection (**RC**) and Unreliable Connection (**UC**) service is associated with a single like QP. Reliable delivery means each QP maintains sequence numbers and acknowledges all messages to ensure that each WR completes exactly once, in order, and without corruption. Unlike Virtual Interface (VI), InfiniBand also supports datagrams. Reliable Datagram (**RD**) service allows a QP to communicate with more than one QP, through End-to-End Contexts (**EEC**). Each node needs one EEC for each node with which it communicates but one or more QPs (in one or more processes in a node) may share the same EEC. Each process needs only one QP. Unreliable Datagram (**UD**) service allows a QP to directly communicate with more than one QP but restricts size of data transfer to the size of a single packet. There are also two optional types of Raw Datagram (**RAW**) data link service intended for tunneling EtherType and IPv6, similar to the UD transport service, but requiring a destination QP to be pre-assigned to the source CA. In addition to data packets that provide service to upper layers, link management packets are exchanged between data packets to exchange flow control credits (the number of data packets that can be sent per virtual lane). Link-level Flow Control Packets (LFCP), which don't need headers because they only flow across a link, indicate the number of blocks sent on a VL since link initialization (FCTBS) and the number of blocks that can be received without buffer overflow (FCCL).

Processes are isolated from each other using Protection Domains (**PD**). QPs, CQs, memory regions, memory windows, and address vectors can only be used with each other if they are created in the same PD. Note that a RD QP is a member of both a PD and a RDD.

User and kernel I/O RD traffic are isolated from each other using at least two Reliable Datagram Domains (**RDD**). RD QPs and EECs can only be used with each other if they are associated with the same RDD. Multiple RD QPs and multiple EECs may be associated with the same RDD.

A memory region is registered to pin the memory, to enable the CA to perform virtual and physical address mapping, to specify access privileges (remote read, remote write, local write), and to bind the region to its protection domain. The consumer uses memory keys returned from registering memory to authorize access to its memory locally (**L_KEY**) and remotely (**R_KEY**). Kernel applications may also register physical addresses to establish an I/O virtual address for a set of not-necessarily contiguous memory pages.

Establishing a communication channel between nodes consists of specifying an **Address Vector** and a QP Number (**QPN**). The Address Vector identifies the source port, the destination port, and the characteristics of the path between them. RC and UC packets use QPN to address a destination QP that validates the source LID/GID. RD packets use EEC Number (**EECN**) to address a destination QP that validates the source LID/GID. UD channels only accept packets containing a queue key (**Q_Key**) that matches the Q_Key in the QP context; both QPs must be configured for the same Q_Key if either QP uses a privileged Q_Key. CAs that support RAW datagrams dedicate one EtherType QP and one IPv6 QP per port so RAW datagram packets only address the destination port.

A fabric can be logically divided into partitions that specify which ports are allowed to communicate. The SM for each subnet in the fabric assigns every port to at least one partition. It is expected that the ports of a node that attach to the same fabric are members of the same partitions. Ports that are in the same partition, identified by a partition key (**P_Key**), may be assigned limited membership so other limited members of the partition (e.g., hosts) cannot communicate with it but can communicate to the full members of the partition (e.g., storage devices). Note that there must be correlation between SMs for multiple subnets to act as a single fabric but InfiniBand does not specify this functionality. For example, vendors or customers must ensure that end nodes in different subnets have the same P_Key if they are to communicate.

>>> Insert Completion Queue information here

InfiniBand specifies *service agents*, which reside on nodes being managed, *class managers* that aggregate and serialize access to service agent information, and *management applications*, that generally reside on hosts, and use the service agents or class managers to manage the fabric.

A manager is any entity that initiates communication using the InfiniBand management framework. The Subnet Manager (**SM**) is a special manager; all other managers are called General Service Managers (**GSM**). Every subnet has one master SM and zero or more standby SMs. There can be multiple GSMs on a node

but only 1 GSM per management class. A management class specifies the methods, attributes, and messages exchanged between a manager and an agent.

An agent is an entity that responds to a manager. The Subnet Management Agent (**SMA**) is a special agent on each node (CA, router, or switch) that responds to the SM. All other agents are referred to as General Service Agents (**GSA**). Example GSAs include Subnet Administration (**SA**), the Communications Manager (**CM**), and the SNMP tunneling agent. Every node also provide agents for performance management and baseboard management. Every CA that supports RC, UC, or RD must also provide a communication management agent. Optionally, a node may choose to provide agents for device management, SNMP tunneling, and vendor-specific or application-specific classes.

A class manager is an entity that takes primary control of a GSA and distributes the collected information to GSMs in order to keep the agent simple. Examples of class managers include the SM for SA and the Configuration Manager (CFM) for DevMgt and BM agents.

Communication between these elements use messages called Management Datagrams (**MAD**) that are sent using the UD service type. All MADs have a 24 byte header and 232 bytes of MAD data as shown in the next figure. The appearance of a MAD at a port implies a behavior by the emitter of the message and a required action, and possibly a response, by the target.

bytes					
0	BaseVersion	MgmtClass	ClassVersion	R	Method
4	Status		ClassSpecific		
8	TransactionID				
12					
16	AttributeID		Reserved		
20	AttributeModifier				
24	Data				
...					
252					

MADs are grouped into management classes according to the type of management activity the messages support (IB has defined 8 MgmtClasses but allows for 256):

- Subnet Management (Subn, 01h - LID routed, 81h - directed routed) to discover, initialize and maintain a subnet.
- Subnet Administration (SubnAdm - 03h) provides the means for managers to obtain information about fabric configuration and operation.
- Performance Management (Perf - 04h) provide facilities for examining fabric performance characteristics.
- Baseboard management (BM - 05h) specifies the means to effect in-band low level system management operations.
- Communication Management (CommMgt - 07h) provide the means to setup and manage communications between a pair of QPs or to identify which QP to use for a certain service.
- Device Management (DevMgt - 06h) specifies the means to determine the kind and location of devices on a fabric.

- SNMP tunneling (SNMP – 08h) specifies mechanisms to support transport of SNMP operation through an InfiniBand fabric.
- There are also vendor-specific (09-0Fh), application-specific (10-1Fh), and reserved (00, 20-80h, 82-FFh) management classes.

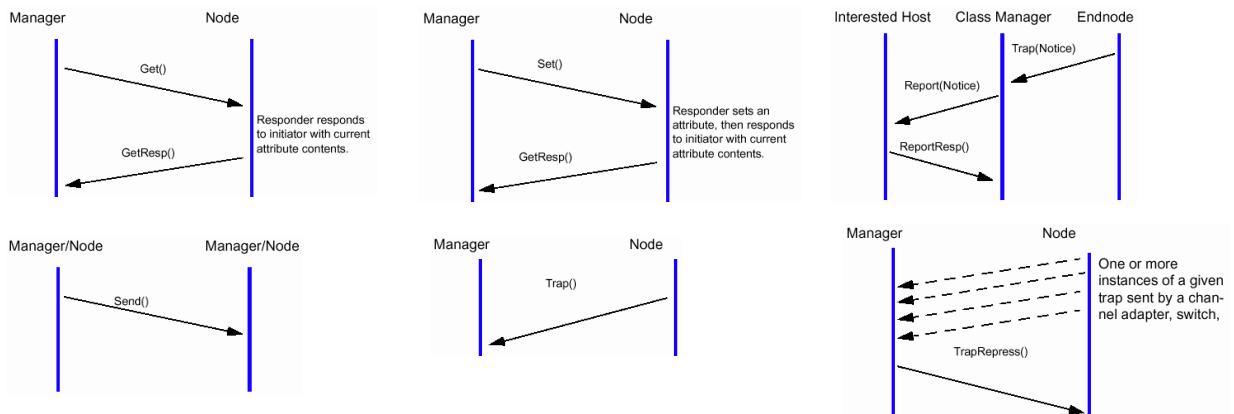
Subnet Management MADs are referred to as Subnet Management Packets (**SMP**). LID routed SMPs are forwarded by switches based on the LID of the destination. Directed route SMPs are forwarded based on a vector of port numbers that define a path through the subnet before LIDs are assigned to the nodes. Other MADs are referred to as General Services Management Packets (**GMP**).

The Subnet Management Interface (**SMI**) is associated with QP0 and VL15 and is used exclusively for communicating SMPs between the SM and the SMAs without enforcement of partitioning or Q_Keys. SMPs destined for a SM are posted to QP0. SMPs destined for a SMA are processed directly by the SMA but may be posted to QP0 as a SM policy.

The General Services Interface (**GSI**) is associated with QP1 and is used exclusively for communicating GMPs to General Services Agents (GSA) using Q_Key 80010000h and normal UD SL/VL/partitioning rules. GMPs processed by GSAs below the verbs interface and GMPs redirected to another QP/port are not posted to QP1. GMPs destined for GSAs above the verbs interface are posted to QP1. GSAs have their own QPs for sending and receiving GMPs; IB does not require a GSM to use the GSI. An application that supports multiple MgmtClasses can share a QP for all of its GSAs.

Attributes are structures, typically representing hardware registers in CAs, switches, or routers, which a management class manipulates. Attributes must be retrieved and modified in their entirety. *ClassPortInfo* (provides node port info), *Notice* (describes node events), and *InformInfo* (provides information for subscribing to a class manager for event forwarding) are attributes common across multiple management classes.

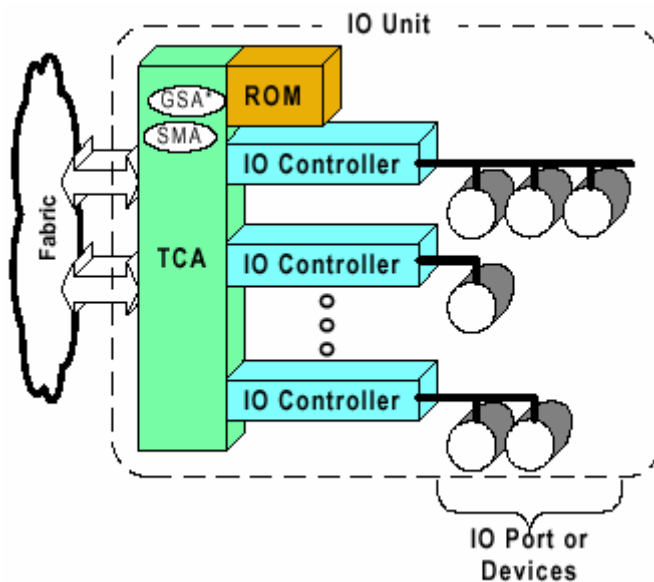
Common methods for exchanging management information are defined: *Get/Set* for accessing attributes, *Traps* for agents to report asynchronous events, *Reports* for class managers to distribute event information, and *Subscriptions* for applications to register their interest in traps and reports. There are three types of methods, messages, requests, and responses as illustrated by the ladder diagrams in the next figure. Messages are *Send()*, *Trap()*, and *TrapRepress()*. Requests are *Get()*, *Set()*, and *Report()*. Responses are *GetResp()* and *ReportResp()*.



The Communications Manager (CM) manages communications for all service types. IB defines a peer-to-peer protocol that a CM uses to exchange information with a CM on another node. CM responsibilities include setting up services, resolving services, and maintaining QPs (including supporting APM). Automatic Path Migration (APM) is an optional facility that enables connection recovery in the case of failures. Automatic path migration is available for RC and UC QP service types and RD EECs.

>>> Insert IORM info here

An InfiniBand I/O Unit (**IOU**), consists of a TCA and one or more I/O Controllers (**IOC**), as depicted in the next figure. A DevMgt class GSA residing in the TCA is used by hosts to discover the number of IOCs (IOUnitInfo attribute) and the properties of each IOC (IOControllerProfile attribute). The host uses this information to match an I/O driver to the IOC. Boot code in boot ROM acts as a ROM repository which loads drivers from option ROM. The I/O driver discovers and manages devices behind the IOC. The I/O driver initiates a service connection, consisting of one or more channels, each using any service type, with the IOC. The I/O driver uses InfiniBand transport services to transfer commands and data, acting as an InfiniBand Upper Level Protocol (ULP). An example ULP is the SCSI RDMA Protocol (SRP).



Question: The SCSI Target is the IOC but Device Management only supports 255 LUNs per IOC.

>>> Insert info on Storage Boot Wire Protocol (SBWP) here
Terms related to InfiniBand addressing are:

Global Identifier (GID) is a 128-bit unicast or multicast identifier used to identify a CA port, a switch, a router port, or a multicast group. A GID is a valid 128-bit IPv6 address (per IETF RFC 2373) with additional restrictions. Each CA port, switch, and router port, is assigned a unicast GID. The SM may assign CA ports and router ports additional unicast GIDs by concatenating a 64-bit GID subnet prefix (so 2^{64} subnets/fabric) with the set of SM-assigned EUI-64 values. A QP in a CA, switch or router may be addressed using the **default subnet prefix (FE80::0h)** as well as an

assigned GID (preferred). A multicast GID is an identifier for a group of CA ports and router ports.

GID 0:0:0:0:0:0:0:0 is reserved. GID 0:0:0:0:0:0:0:1 is used for IPv6 loopback. A unicast GID is created by concatenating the GID prefix to a EUI-64 identifier, such as a GUID. A unicast GID begins with FEh and may have link-local, site-local, or global scope. A unicast link-local GID prefix uses the default subnet prefix. A unicast site-local GID is unique within a collection of subnets identified by the last 2 bytes of the GID subnet prefix. A unicast GID with global scope uses all 64-bits to uniquely identify the subnet. A multicast GID begins with FFh, followed by four flag bits and four scope bits, followed by a 14 byte multicast identifier. The flag indicates whether the GID is well-known as defined in RFCs 2373 and 2375 (0h) or transient (1h). The scope indicates whether the multicast address scope is link-local (2h), site-local (5h), organization-local (8h), or global (Eh). The multicast link-local GID of FF02::1 is used to broadcast to all multicast-capable CAs within a subnet.

Globally Unique Identifier (GUID) is a globally unique EUI-64™ compliant identifier. Each CA, CA port, Switch, Router, and Router port are assigned a GUID by their manufacturer with the universal/local bit set to one to indicate global scope. A SM may assign additional EUI-64 identifiers with the universal/local bit set to zero to indicate local scope.

A boot device persistent identifier may be created consisting of the Node GUID of the IO unit, the GUID of the IO controller, and a device ID (a 16-bit value assigned by the manufacturer). The following attributes from boot information records are recommended to gracefully handle hot-swap scenarios: chassis GUID, chassis name, module ID (location of IOU within chassis), node (IOU) name, and IOC ID (location of IOC within IOU).

Handles are used by applications as *local* identifiers for Channel Adapters, Protection Domains, Memory Regions (L_Key is also used by hardware), Memory Windows, Address Vectors, Completion Queues, Queue Pairs, Reliable Datagram Domains (referred to as an object rather than with a handle), and End-to-End Contexts (EEC Number is also used by hardware). A handle is unique within its scope. An application uses LIDs/GIDs, R_Keys, 24-bit EEC Numbers, and 24-bit QP Numbers (so 2**24 QP/CA) are used as *remote* identifiers for CAs, Memory Regions/Windows, EECs, and QPs respectively. An application is not aware of PDs, RDDs, EECs, Address Vectors, or CQs, on other nodes.

IEEE 64-bit extended unique identifier (EUI-64™) is a concatenation of a high order 24-bit company identifier assigned by the IEEE to a low order 40-bit extension identifier assigned by the manufacturer, with the restriction that the high order 16 bits of the extension identifier cannot be FFFFh or FFFEh which are used to support encapsulation of MAC-48 and EUI-48™ values respectively.

IPv6 addresses are usually written as eight groups of four hex digits, with each group separated by a colon. Leading zeros may be omitted and consecutive zeros can be abbreviated by a single double colon "::" once in the address. The high order "netbits" identify the network and the low order "hostbits" identify a host or subnetwork. The number of netbits used are indicated after a slash "/".

Local Identifier (LID) is a 16-bit identifier unique within a subnet used to identify a CA port, a switch, a router port, or a multicast group (2**16 less reserved and

multicast addresses leaves about 48K nodes per subnet). LID 0000h is reserved. A unicast LID ranges from 0001h to BFFFh. A multicast LID ranges from C000h to FFFFh. LID FFFFh indicates a packet is destined for QP0 on the port that received it; this "permissive LID" is only used for subnet management before LIDs are assigned. A source LID (SLID) must be a unicast LID. A unicast destination LID (DLID) may be the LID of a CA port, a router port (if the destination node is not in the same subnet), or switch port 0. A multicast DLID refers to one or more ports within a subnet participating in the multicast group. A port may be the target of zero or more multicast flows. A SM assigns each CA port, switch port 0, and router port a range of LIDs by assigning a base LID and an LMC value, indicating that 2**LMC LIDs are assigned beginning from the base LID.

LID Mask Control (LMC) is a 3 bit field indicating the number of low order bits (referred to as path bits) which may be ignored when validating a destination address. The path bits may be changed to vary the path through the subnet. A switch only needs one LID so the LMC is 0.

Message Sequence Number (MSN) is a 24-bit value used as a hint to a requester to determine which messages the responder has completed. MSNs only apply to reliable service types (RC and RD).

Packet Sequence Number (PSN) is a 24-bit value in every packet generated by the requester and returned by the responder. The requester uses it to detect missing responses. The responder uses it to identify a packet as being new, duplicate, or invalid.

Path is the set of links, switches, and routers a packet traverses from a source to a destination. Within a subnet, a path is defined by the source LID, the destination LID, and the Service Level.

Service Identifier (SID) is a 64-bit value in CM messages used to associate incoming requests with a service provider.

Service Name is a 40-byte string used to identify a particular service. For example, an IOC acting as an SRP target port registers a Service Name of 'SRP.T10.NCITS'.

Service Levels (SL) are 4 bit values. A Traffic Class (TC) is an 8 bit value that communicates a global end-to-end class of service.

Subnet Prefix is an identifier of up to 64 bits used to uniquely identify a set of links, CA ports, and switches which are managed by a common SM.

Virtual Addresses are up to 64 bits. A Data Segment is defined by a Virtual Address, an L_Key, and a Length.

Virtual Lanes (VL) are 4 bit values.

InfiniBand uses the following keys to authenticate requests:

Baseboard Management Key (B_Key) is an 8 byte MAD value that authenticates that a source is allowed to perform the requested operation.

Local Key (L_Key) is a 32-bit value?

Management Key (M_Key) is a 64-bit value.

Partition Key (P_Key) is a 16-bit value carried in packets used to validate the sender's right to communicate with a destination port. A table in each port contains the partition keys the SM assigned to that port. The high-order bit of a P_Key identifies whether the port is considered a full or limited member of a partition. All members can accept packets from full members but only full members can accept packets from limited members.

Queue Key (Q_Key) is a 32-bit value stored in the QP context and carried in datagrams used to validate the sender's right to communicate with the specified receive queue. This is the only key under OS/application control; all others are owned by the SM.

Remote Key (R_Key) is a 32-bit value provided by an HCA when a consumer registers memory and carried in RDMA packets used to validate hardware access to memory.

Request Identifier is a 32-bit value.

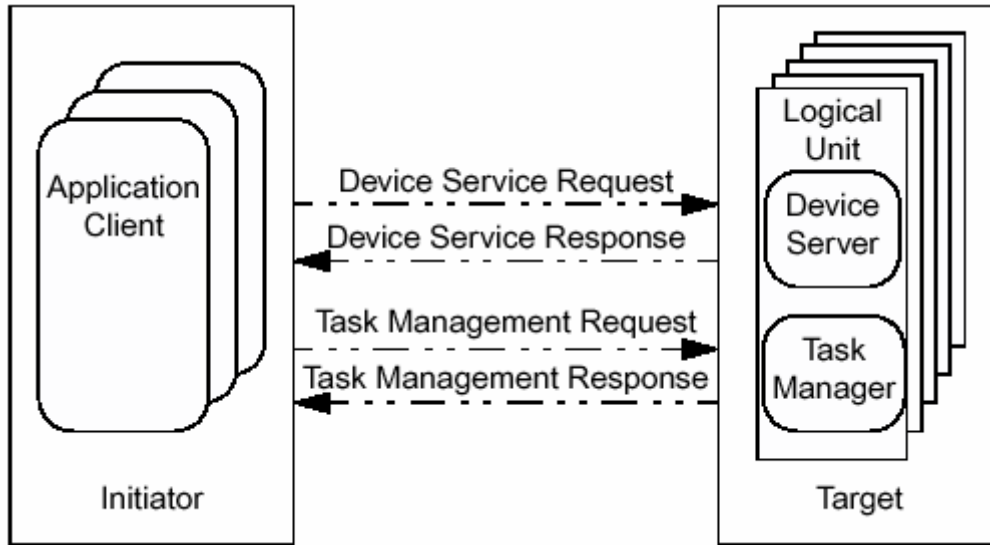
Subnet Manager Key (SM_Key) is a 64-bit value used to protect SMInfo access.

Transaction Identifier (TID) is a 64-bit value must be the same for every MAD in any request, response, or series of Sends that corresponds to a single operation. The *combination* of TID, source GID, and MgmtClass must be different from that of any other currently executing operation.

Work Request Identifier is a 64-bit value.

SCSI Architecture (SAM-2)

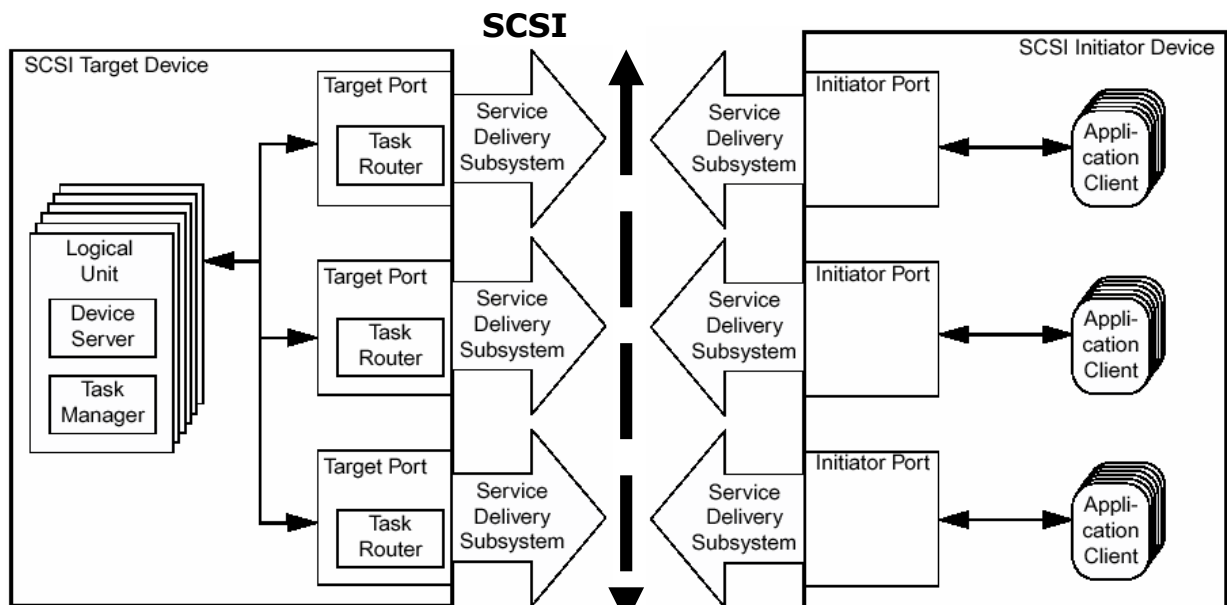
In the Small Computer System Interface Architecture, a client originates requests and a server responds to them as shown in the next figure. SCSI commands are addressed to logical units, which contain a device server to process SCSI commands and a task manager to sequence and process task management requests. When a device containing at least one port originates SCSI commands and task management requests it is called an initiator. When a device containing at least one port contains logical units it is called a target. A SCSI domain contains at least one SCSI device, one initiator port, and one target port, interconnected by a service delivery subsystem through which application clients and device servers communicate. A target port contains a task router that sends tasks to logical units; if the LU is unknown the task is sent to LUN 0; if no LU is specified, the task management function is broadcast to all LUs known to the router. In I/O operation is a SCSI command, a series of linked SCSI commands, or a task management function.



An application client, that is, a thread of processing within an initiator device such as a device driver, sends one or more SCSI commands to a target device. Requests, represented as tasks, are processed by the device server or task manager within the logical unit. A logical unit contains one or more task sets, each of which contain zero or more tasks. A device server may reject commands that are not sent from the set of initiators determined by the application client using reservation commands.

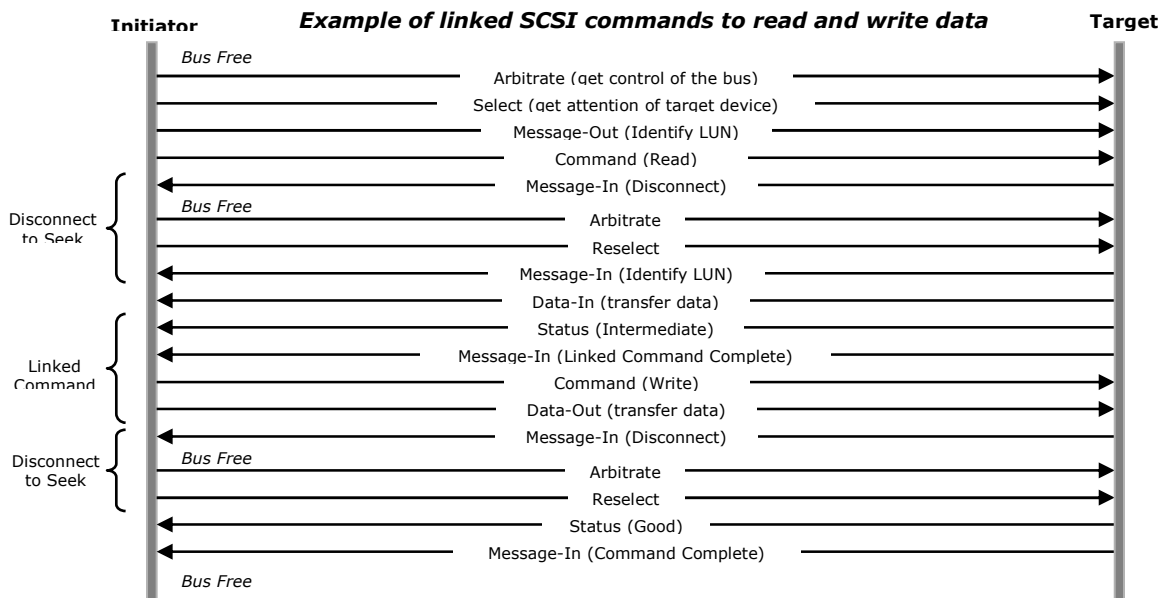
A logical unit may contain other logical units, referred to as dependent logical units. A device server that implements the hierarchical structure for dependent logical units sets the HISUP bit in the standard INQUIRY data returned by logical unit 0 (see SPC-2).

A SCSI device can have one or more ports as depicted in the following figure. Note that an initiator does not know if different target ports are in different devices. Likewise, a target does not know if different initiator ports are in different devices. However, application clients can discover that a logical unit is accessible via multiple target ports using the INQUIRY command vital product data page.



An application client sends requests by executing a remote procedure with eight input parameters: a nexus, a command descriptor block (CDB), a task attribute, number of bytes to transfer into the input buffer, an output buffer containing command-specific information, number of bytes to transfer from the output buffer, an option to automatically return sense data, and a command reference number. Output is returned in a buffer to hold command-specific information returned by the logical unit, a buffer to hold autosense data, and completion status.

A command executes in subsequent parts called phases, which may be interleaved with phases from other commands to other devices. Some phases are missing in some commands but the general sequence is arbitration, selection (with attention), message-out (initiator identifies the LUN), command (initiator sends the CDB), data-in (target sends data to the initiator) or data-out (initiator sends data to the target), status (target reports status), and message-in (target reports command completion).



The CDB defines the operation to be performed by the device server. A CDB may have a fixed length of 6, 10, 12, or 16 bytes, as shown in the next figure, or a variable length, as shown in the figure after that, of between 12 and 260 bytes. The operation code is the first byte in all CDB formats. The control byte is the last byte in fixed length CDB formats and the second byte in variable length CDB formats.

Fixed length CDB format: (see SAM-2)

Bit Byte	7	6	5	4	3	2	1	0
0	GROUP CODE			COMMAND CODE				
1	Command specific parameters							
n-1	Command specific parameters							
n	Vendor specific		Reserved			NACA	Obsolete	LINK

Variable length CDB format: (see SPC-2)

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Fh)							
1	CONTROL							
2	Reserved							
3	Reserved							
4	Reserved							
5	Reserved							
6	Reserved							
7	ADDITIONAL CDB LENGTH (n-7)							
8	(MSB)	SERVICE ACTION						(LSB)
9								
10	Service action specific fields							
n								

The INQUIRY command (operation code 12h), processed by all SCSI device servers, requests standard device information, vital product data (VPD), or information about which commands are supported by the device server.

Standard INQUIRY data includes ASCII data identifying the manufacturer (8 bytes), the product (16 bytes), the product revision level (4 bytes), and hex data identifying up to eight standards to which the device claims conformance (in 2 byte fields, such as 0900h for FCP-2, 0940h for SRP, and 0960h for iSCSI). Note that T10 maintains a list of SCSI Vendor ID assignments at <http://www.t10.org/lists/vid-alpha.htm>.

The application client requests VPD data by setting the EVPD bit to one and specifying the page code of the desired VPD data. The supported VPD pages (code 00h) and device identification page (code 83h) are mandatory. Optionally, the INQUIRY command may support the unit serial number page (code 80h), the target operating definition page (code 82h), the FRU information page (codes 01h-7Fh), and vendor-specific pages (codes C0h-FFh). The unit serial number page returns a vendor-assigned serial number for a target or logical unit.

The application client requests command support data by setting the CMDDDT bit to one and specifying the SCSI operation code of the desired CDB. Information returned indicates if the requested operation is supported and if so, if it is implemented in conformance with a SCSI standard or in a vendor-specific manner. However, as this does not support VL CDBs or service actions, the REPORT SUPPORTED OPERATION CODES command (operation code A3h) has been proposed as a more comprehensive method to determine which commands a logical unit supports.

The REPORT LUNS command (operation code A0h) requests a list of the logical unit numbers of all logical units connected to the device. This inventory may also include logical unit numbers for vendor-specific logical units. A device supporting multiple LUN addresses must support a REPORT LUNS command addressed to LUN 0. Command support is optional for a device having a single logical unit or by logical units other than zero.

SCSI addressing information:

Name is unique within a specified context and does not change. Also called a world-wide identification (**WWID**).

SCSI Identifier represents either an initiator port or a target port identifier. Also called device identifier and port identifier. Note that other standards define the value of SCSI identifiers. For example, SPI-2 defines target identifiers to be in the range 0-7, 0-15, and 0-31.

Task Identifier, assigned by a device server, is an initiator identifier, a logical unit number, and if the task is tagged, a tag. A task identifier in use is unique if one or more of its components is unique.

Tag contains up to 64 bits, assigned by an initiator to ensure the unique identity of a tagged task within a single task set. A tagged task also includes one of the task attributes (simple, ordered, head of queue, or auto contingent allegiance) that allows an initiator to specify tagged task processing relationships.

Nexus is the relationship between a SCSI initiator port, a SCSI target port, optionally a logical unit, and optionally a task.

Identification Descriptor identifies a device or port with the same descriptor when accessed through any path. The INQUIRY device identification page (code 83h) is used to retrieve identification descriptors. Logical units may have more than one identification descriptor if several identifier types are supported. Identifier types may be vendor-specific (not guaranteed to be globally unique), use an 8 byte Vendor ID prefix (vendor guarantees uniqueness), comply with the IEEE 64-bit Global Identifier standard (EUI-64), or be a FC-FS Name Identifier (WWN). Port descriptors are 4 byte binary relative identifiers from 1h ("port A") to 7FFFFFFh ("port 2 147 483 647").

Logical Unit Identifier (LUID) uniquely identifies a logical unit in a SCSI domain.

Logical Unit Number (LUN) is a 64 bit structure that allows up to 4 levels of devices to be addressed under a single target, as shown in the following figure. Each level uses 2 bytes to address the devices on that level. The high order 2 bits indicate the address method (00b = peripheral device, 10b = logical unit, 01b = device type specific) and the remaining 14 bits are address method specific. When a command is sent to the target, the LUN is adjusted to create a new LUN by shifting each 2 byte addressing field up a level, filling in with zeros. Targets that contain 256 or fewer logical units use a single level LUN structure, meaning all fields are 0b except byte 1.

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)	FIRST LEVEL ADDRESSING						(LSB)
1		SECOND LEVEL ADDRESSING						(LSB)
2	(MSB)	THIRD LEVEL ADDRESSING						(LSB)
3		FOURTH LEVEL ADDRESSING						(LSB)
4	(MSB)							(LSB)
5								(LSB)
6	(MSB)							(LSB)
7								(LSB)

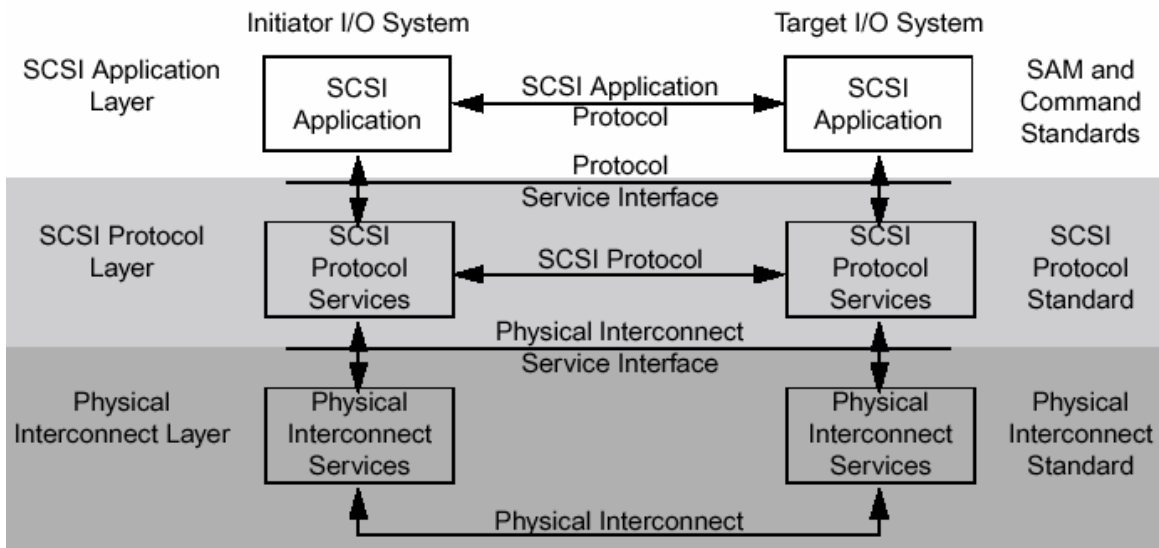
The peripheral device address method, shown in the next figure, relays commands to LUN 0 of the target identified by the 1 byte target/LUN field. LUN 0, used to determine information about a target and its logical units, always uses this address method. A bus identifier of zero indicates a logical unit at the current level. A bus identifier from 1 to 63 represents a bus that physically connects a group of devices to the current level device. If the range of possible target identifiers is too large to fit in 1 byte, the target/LUN field may contain a mapped representation of the target identifier.

Bit Byte	7	6	5	4	3	2	1	0
n-1	0	0	BUS IDENTIFIER					
n	TARGET/LUN							

The logical unit address method, shown in the following figure, relays commands to the LUN within the target located on bus number as indicated. If the range of possible target identifiers is too large to fit in 5 bits, the target field may contain a mapped representation of the target identifier.

Bit Byte	7	6	5	4	3	2	1	0
n-1	1	0	TARGET					
n	BUS NUMBER			LUN				

SCSI defines an application layer, a transport protocol layer, and a physical interconnect layer as shown in the next figure. The subsystems that make up the protocol and interconnect layers are collectively referred to as the service delivery subsystem. A client sends a request to a server that returns a response at the application layer. The application layer uses the services of the protocol layer to implement this transaction. At the client-side, the upper layer protocol (ULP) uses a *Protocol service request* to invoke a service provided by the lower layer protocol (LLP). At the server-side, a *Protocol service indication* from the LLP to a ULP signals that an asynchronous event has occurred. The server-side ULP calls the LLP to respond to the indication using a *Protocol service response*. At the client-side, a *Protocol service confirmation* sent from the LLP to the ULP signals that the request has completed.



Example SCSI Application Protocols are defined in the SPC-3 shared command set and device type specific command sets such as SBC-2 (disk), SCC-2 (RAID controller), SSC-2 (tape), and SES (enclosure).

Example SCSI Transport Protocols are defined in FCP-2, SSA-S3P, SRP, and iSCSI. Example SCSI Physical Interconnects are defined in FC-FS, FC-AL-2, SPI-4 (parallel SCSI), SSA-PH-2, VI, InfiniBand, and Gigabit Ethernet.

Fibre Channel Protocol (FCP-2)

FCP (an FC-4) maps SCSI (a ULP) I/O operations defined by SAM-2 into a Fibre Channel Exchange using FC-FS services. A layered communication model similar to the SCSI request, indication, confirmation, response model is provided by FC-FS. The FC-4 generates a *FC_FS_SEQUENCE.request* to define a transfer of one or more ULP data blocks. The source FC-FS may return a *FC_FS_SEQUENCE_TAG.indication*, then segments the data blocks into frames and transmits the frames as a single Sequence. After receiving the frames, the destination FC-FS reassembles the ULP data blocks, issues acknowledgements as appropriate, and generates a *FC_FS_SEQUENCE.indication* to define the transfer of the completed Sequence to the appropriate destination FC-4. After Sequence delivery has been completed, the source FC-FS may issue *FC_FS_SEQUENCE.confirmation* to indicate success or failure of the request.

FCP is described in terms of Information Units (IUs) and Exchanges generated by a pair of FCP_Ports. Each IU is contained in a single Sequence and each Sequence may only carry a single IU. Up to 65,35 Exchanges and up to 256 active Sequences may be open simultaneously between an initiator and a target FCP_Port. However, some Exchange IDs and at least one extra Sequence ID should always be available for task management.

IUs sent to targets:

- FCP_CMND carries either a SCSI Command or a task management request.
- FCP_CONF is used to confirm the receipt of a FCP_RSP.

IUs sent to initiators:

- FCP_XFER_RDY indicates that the target is prepared to receive part or all of the data for a write command.
- FCP_RSP provides completion information for FCP I/O operations.

IUs sent to both initiators and targets:

- FCP_DATA is used to transfer data in the same Exchange that sent the FCP_CMND requesting the transfer. If more than one IU is used to transfer the data, the relative offset is used to ensure that the SCSI data is reassembled in the proper order.

FCP Information Units are depicted in the following table.

Field	Bits	Value	Notes
<i>FCP_CMND (28 bytes + lengths of additional CDB and Data Buffer Size)</i>			
FCP_LUN	64		SCSI command or task management request LU
Command Reference Number (CRN)	8		Confirms receipt and ordering of commands
Reserved	5		
Task Attribute	3		000b=Simple, 001b=Head of Queue, 010b=Ordered, 100b=ACA, 101b=Untagged
obsolete	1		<i>Task Management Flag</i> byte indicates task management requests using a new exchange:
CLEAR ACA	1		
TARGET RESET	1		
LOGICAL UNIT RESET	1		
Reserved	1		
CLEAR TASK SET	1		
ABORT TASK SET	1		
reserved	1		
Additional CDB Length	6	x	x is a multiple of 4 byte words
RDDATA	1		Initiator expects <i>FCP_DATA</i> IUs for SCSI Read
WRDATA	1		Initiator expects <i>FCP_DATA</i> IUs for SCSI Write
FCP_CDB	128		CDB to be sent to addressed LU
Additional FCP_CDB	4*x*8		Used when CDB is greater than 16 bytes
FCP_DL (Data Buffer Size)	32		Maximum bytes to be transferred to/from buffer
<i>FCP_XFER_RDY (12 bytes)</i>			
FCP_DATA_RO	32		SAM-2 application client buffer offset which may be used by target to request out of order write data if allowed by FMDP disc-recon page
FCP_BURST_LEN	32		SAM-2 data delivery request byte count used to request the initiator to send IU of this length
Reserved	32		
<i>FCP_DATA (transmitted in the same exchange that sent the FCP_CMND requesting the transfer)</i>			
<i>FCP_RSP (24 bytes plus length of sense and response data)</i>			
Reserved	83		Completion information for FCP I/O operations
FCP_CONF_REQ	1		Initiator to send <i>FCP_CONF</i> to confirm <i>FCP_RSP</i>
FCP_RESID_UNDER	1		<i>FCP_RESID</i> expected bytes were not transferred
FCP_RESID_OVER	1		<i>FCP_DL</i> too short to hold <i>FCP_RESID</i> bytes
FCP_SNS_LEN_VALID	1		Examine <i>FCP_SNS_INFO</i> for possible error
FCP_RSP_LEN_VALID	1		Examine <i>FCP_RSP_INFO</i> for possible error
SCSI Status Code	8		SAM-2 Status code (if <i>FCP_RSP_LEN_VALID</i> =0b)
FCP_RESID	32		Number of bytes that were not transferred
FCP_SNS_LEN	32	n	Number of bytes in <i>FCP_SNS_INFO</i> field
FCP_RSP_LEN	32		Number of bytes in <i>FCP_RSP_INFO</i> field
FCP_RSP_INFO	64		Next 8 bytes is <i>FCP_RSP_INFO</i> field if valid
Reserved	24		
RSP_CODE	8		Protocol failure information: 00h=Task Management function complete, 04h=Task Management function rejected, 05h=Task Management function failed, 01h= <i>FCP_DATA</i> length different than <i>FCP_BURST_LEN</i> , 02h= <i>FCP_CMND</i> fields invalid, 03h= <i>FCP_DATA</i> parameter does not match <i>FCP_DATA_RO</i>
Reserved	32		
FCP_SNS_INFO	n		SPC-2 autosense data
<i>FCP_CONF (no payload - confirms receipt of FCP_RSP if supported and requested)</i>			

The address of each FCP_Port is defined by its address identifier. Each FCP I/O operation is identified by the FCP I/O operation's fully qualified exchange identifier (FQXID). The FQXID is composed of the initiator address identifier, the target address identifier, the OX_ID and the RX_ID. Addressability of logical units uses the logical unit number provided in the FCP_CMND IU. Subsequent identification of the FCP I/O operation and the Exchange which carries the protocol interactions for the FCP I/O operation uses the FQXID. FCP devices do not use the Process_Associator.

The target uses the OX_ID, and, if it has been assigned, the RX_ID to perform error recovery and task management functions. The task retry identifier is used as a supplemental task identifier if task retry identification is supported and enabled.

An initiator completes a Process Login (PRLI) with a target before exchanging FCP IUs to identify the capabilities that the Originator FCP_Port expects to use with the Responder FCP_Port and to determine the capabilities of the Responder. Each target has knowledge of the Port Name of each initiator through the FC login process. If a target receives a PRLI or an N_Port Login (PLOGI) from an initiator FCP Port with a previously known WWN but with a changed initiator identifier, the device server shall assign the new initiator identifier to the existing registration and reservation to the initiator port having the same WWN. Each logical unit shall be able to present a WWN through the INQUIRY command vital product data device identification page.

SCSI RDMA Protocol (SRP)

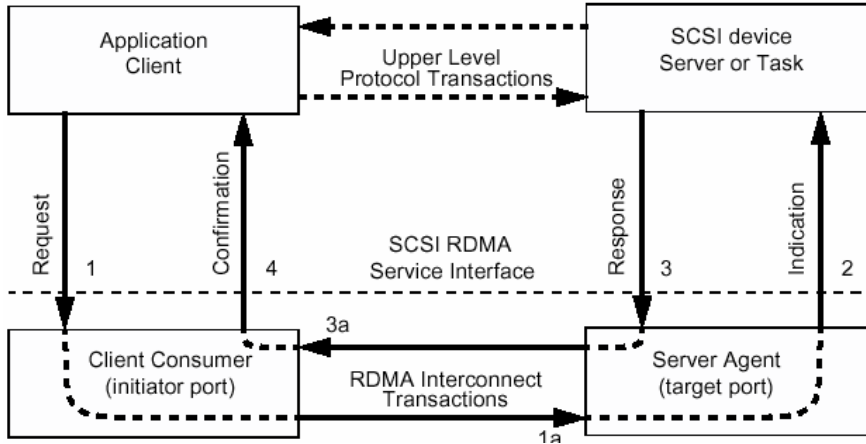
SRP is a transport protocol that allows SCSI command set information to be transmitted using an RDMA communication service between pairs of consumers using messages for control information and RDMA operations for data transfers. Virtual Interface (VI) Architecture and InfiniBand interfaces support such RDMA semantics. A client consumer requests that the RDMA communication service establish an RDMA channel. The request is directed to a server and, if successful, resolved to a server consumer. The resulting RDMA channel provides communication between the client consumer and the server consumer. Multiple RDMA channels may be established between an initiator port and a target port (I_T nexus). Though each SRP request is confined to a single RDMA channel, SCSI tasks and task management functions may use more than one RDMA channel associated with the same I_T nexus.

If the RDMA communication service cannot deliver a message it will disconnect the RDMA channel. Each message is delivered exactly once, complete and error-free, and in the same order as they were sent by the same consumer on the same RDMA channel. Data from RDMA Write operations requested before a message is sent by the same consumer on the same RDMA channel will be available to the receiving consumer prior to delivery of a message. Messages sent on different RDMA channels or by different consumers may be delivered in any order. RDMA Read operations may complete in any order.

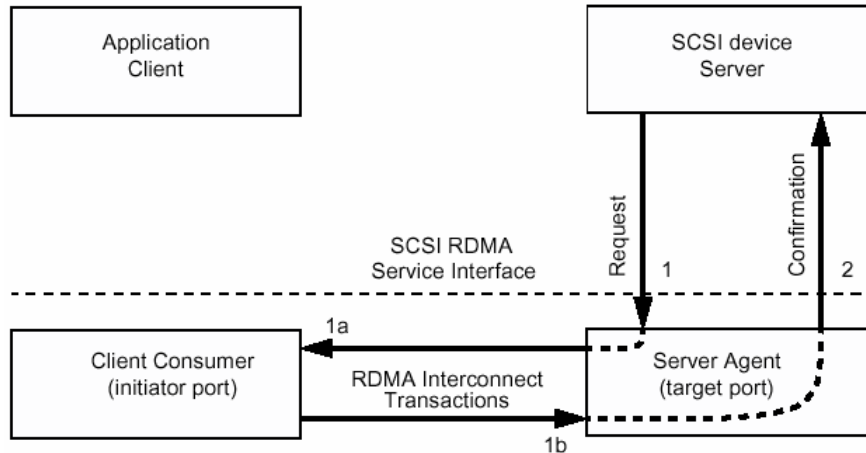
SRP Initiators must support SEND, RDMA Write (to), and RDMA Read (to). SRP Targets must support SEND, RDMA Write (from), and RDMA Read (from). SRP does not use RDMA Write with immediate data or RDMA Atomic operations. Also note that every CA that supports SRP must provide a CM agent since SRP uses RC.

SRP provides services to enable an application client to request and manage tasks and to enable a device server to receive commands and move data to and from an application client.

Application clients use the four step confirmed service as shown in the following figure.



SCSI device servers use the two step confirmed service as shown in the following figure.



Each SRP information unit (IU) is classified as a SRP request (that conveys SCSI commands, task management requests, and RDMA channel management requests) or a SRP response (sent on the same RDMA channel as the request). An SRP request communicates the initiation of a remote procedure call; the corresponding SRP response communicates the remote procedure call's completion. The first byte of each IU uniquely identifies an IU and its format. A 64-bit tag value (bytes 8-15) in each IU provides a way to match SRP requests with their corresponding responses. The tag is unique among all of the requestor's outstanding requests with a particular responder.

An initiator may send a request when a channel's *Request Limit* is greater than zero. The initiator decrements a channel's *Request Limit* by one whenever it sends a request on that channel. The target adds the value of *Request Limit Delta* to a channel's *Request Limit* whenever it receives an IU on that channel. SRP targets have at most one outstanding request per RDMA channel.

Requests sent from initiators to targets are SRP_LOGIN_REQ (type 00h), SRP_TSK_MGMT (01h), SRP_CMD (02h), and SRP_I_LOGOUT. The responses are SRP_LOGIN_RSP (C0h), SRP_LOGIN_REJ (C2h), and SRP_RSP (C1h). The maximum IU size sent from an initiator to a target is 64 bytes. SRP_TSK_MGMT requests specify a 64-bit LUN as defined in SAM-2 and a 1 byte code indicating the task to be performed (ABORT TASK = 01h, ABORT TASK SET = 02h, CLEAR TASK SET = 04h, LU RESET = 08h, CLEAR ACA = 40h). An SRP_CMD request conveys an LU (the address of the logical unit of the I_T_L_Q nexus for the current task), a CDB, and buffer descriptors.

Requests sent from targets to initiators are SRP_T_LOGOUT (type 80h), SRP_TPAR_REQ (81h - target parameter), SRP_AER_REQ (82h - asynchronous event report). The responses are SRP_TPAR_RSP (41h) and SRP_AER_RSP (42h). The maximum IU size sent from a target to an initiator is 52 bytes (except for AERs).

Three SCSI mode pages influence, control, or report SRP behavior: disconnect-reconnect, port control, and LU control. The port and LU control mode pages contain those parameters that select target port or logical unit operation options but are not currently defined for SRP devices. The disconnect-reconnect page provides the application client the means to tune the service delivery subsystem:

- *Maximum Burst Size* is a 16-bit value that indicates the maximum number of 512 byte blocks (0 = unlimited) that a device server shall read or write with RDMA operations. A router between an SRP device and another protocol device (e.g. FCP) may intercept and adjust this value to reflect its own maximum buffering capabilities.
- *Enable Modify Data Pointer* is a bit that indicates whether a target may issue RDMA's for a single SCSI command in any order (1b) or if the target must generate continuously increasing RDMA addresses for a single SCSI command.
- The *Fair Arbitration*, *Disconnect Immediate*, *Data Transfer Disconnect Control*, and *First Burst Size* fields are reserved for SRP devices.

A device server provides the Data-in and Data-out delivery services to transfer data to or from the initiator port. RDMA-Write is used for Data-in delivery such as a read from disk. RDMA-Read is used for Data-out delivery such as writing data to disk. If both Data-in and Data-out delivery are used to process a single command, the device server combines both responses into a single service response.

A device server supplies an RDMA channel, a memory handle, a range of addresses, and either data to be written into the specified range of addresses for an RDMA-Write, or a buffer into which to place the data read from the specified range of addresses, for RDMA-Read.

The SRP_TASK_MGMT request with the appropriate *Task Management Flags* set, indicate which SCSI Task Management function is to be sent to a SCSI device. The functions supported by SRP are Abort Task, Abort Task Set, Clear Task Set, Clear ACA, and Logical Unit Reset; Target Reset and Wakeup are not supported.

SRP Login and Logout IUs are listed in the following table.
(note: Ask Ed G. if SRP_LOGIN_REJ and SRP_T_LOGOUT should both use type 80h.)

Field	Bits	Value	Notes
<i>SRP LOGIN REQ (64 bytes)</i>			
Type	8	00h	Login Request sent from I to T
Reserved	56		
Tag	64		Requestor assigns unique identifier for outstanding SRP requests with responder
Requested Maximum I-T IU	32		At least 64 bytes
Reserved	45		<i>Required buffer formats</i> field is bytes 24-25
Indirect Data Buffer Descriptor	1		Indirect buffers may be used (1b) or not (0b)
Direct Data Buffer Descriptor	1		Direct buffers may be used (1b) or not (0b)
Reserved	7		
Multi Channel Action	2		00b=Terminate, 01b=Independent
Reserved	40		
Initiator Port Identifier	128		I T Nexus associated with this channel
Target Port Identifier	128		I T Nexus associated with this channel
<i>SRP LOGIN RSP (52 bytes)</i>			
Type	8	C0h	Login successful response sent from T to I
Reserved	24		
Request Limit Delta	32		Initiator adds to Request Limit (flow control)
Tag	64		Uniquely identifies outstanding SRP request
Maximum I-T IU	32		At least Requested Maximum I-T IU
Maximum T-I IU	32		At least 52
Reserved	13		<i>Supported buffer formats</i> field is bytes 24-25
Indirect Data Buffer Descriptor	1		Indirect buffers may be used (1b) or not (0b)
Direct Data Buffer Descriptor	1		Direct buffers may be used (1b) or not (0b)
Reserved	7		
Multi Channel Result	2		00b=No channels, 01b=Channels terminated 10b=Channels independent, 11b=Reserved
Reserved	200		
<i>SRP LOGIN REJ (32 bytes)</i>			
Type	8	80h ?	Login failure response sent from T to I
Reserved	24		
Reason	32		10h=No reason, 11h=Insufficient resources, 12h=Requested Max I-T IU too large, 13h=Unable to associate channel / nexus, 14h=A requested descriptor is not supported, 15h=Multiple channels not supported
Tag	64		Uniquely identifies outstanding SRP request
Reserved	77		<i>Supported buffer formats</i> field is bytes 24-25
Indirect Data Buffer Descriptor	1		Indirect buffers may be used (1b) or not (0b)
Direct Data Buffer Descriptor	1		Direct buffers may be used (1b) or not (0b)
Reserved	49		
<i>SRP I LOGOUT (16 bytes)</i>			
Type	8	03h	Initiator Logout notification sent from I to T
Reserved	56		
Tag	64		Uniquely identifies outstanding SRP request
<i>SRP T LOGOUT (16 bytes)</i>			
Type	8	80h ?	Target Logout or Channel Failure notification sent from T to I
Reserved	24		
Reason	32		0=No reason, 1=Inactive channel, 2=Invalid IU Type, 3=No corresponding request outstanding, 4=Channel disconnected, 6=Unsupported Data-out format code, 7=Unsupported Data-In format code, 8=Invalid Data-out count, 9=Invalid Data-In count
Tag	64		Uniquely identifies outstanding SRP request

SRP Task Management, Command, and Response IUs are depicted in the next table.

Field	Bits	Value	Notes
<i>SRP TASK MGMT (48 bytes)</i>			
Type	8	01h	SCSI Task Management sent from I to T
Reserved	56		
Tag	64		Uniquely identifies outstanding SRP request
Reserved	32		
Logical Unit Number	64		
Reserved	16		
Task Management Flags	8		01h=Abort Task, 02h=Abort Task Set, 04h=Clear Task Set, 08h=LU Reset, 20h=Restricted, 40h=Clear ACA, Other values=Reserved
Reserved	8		
Tag of task to be managed	64		
Reserved	64		
<i>SRP CMD (48 bytes + lengths of additional CDB and Data-In/Out Buffer Descriptors)</i>			
Type	8	02h	SCSI Command sent from I to T
Reserved	32		
Data-Out Buffer Descriptor Format	4		0h=no Descriptor, 1h=Direct, 2h=Indirect
Data-In Buffer Descriptor Format	4		Same format code value as for Data-Out
Data-Out Buffer Descriptor Count	8		
Data-In Buffer Descriptor Count	8		
Tag	64		Uniquely identifies outstanding SRP request
Reserved	32		
Logical Unit Number	64		
Reserved	13		
Task Attribute	3		000b=Simple, 001b=Head of Queue, 010b=Ordered, 100b=ACA, Other=Reserved
Reserved	8		
Additional CDB Length	6	x	
Reserved	2		
CDB	128		
Additional CDB	4*x		
Data-out Buffer Descriptor			Direct data buffer descriptors are 16 bytes
Data-in Buffer Descriptor			Indirect descriptors are 20+16*count bytes
<i>SRP RSP (36 bytes or minimum length to hold response data or sense data)</i>			
Type	8	C1h	Response to SRP TASK MGMT or SRP CMD
Reserved	24		
Request Limit Delta	32		Initiator adds to <i>Request Limit</i> (flow control)
Tag	64		Uniquely identifies outstanding SRP request
Reserved	18		
DIUNDER	1		<i>Data-in residual count</i> is valid (1b) or not (0b)
DIOVER	1		<i>Data-in residual count</i> is valid (1b) or not (0b)
DOUNDER	1		<i>Data-out residual count</i> is valid (1b) or not (0b)
DOOVER	1		<i>Data-out residual count</i> is valid (1b) or not (0b)
SNSVALID	1		<i>Sense Data List Length</i> is valid (1b) or not (0b)
RSPVALID	1		<i>Response Data List Length</i> is valid (1b) or not
Status	8		As defined by SAM-2 (invalid if RSPVALID=1b)
Data-out residual count	32		Bytes not transferred from data-out buffer
Data-in residual count	32		Bytes not transferred to data-in buffer
Sense Data List Length	32	n	n is a multiple of 4
Response Data List Length	32	m=4	Since <i>Response Data List</i> has only 1 entry
Reserved	24		Each entry in the <i>Response Data List</i> is 4 bytes
RSP_CODE	m*8		SRP_TSK_MGMT request completion status : 00h=Complete, 02h=Invalid request, 04h=Not supported, 05h=Failed, Other values=Reserved
Sense Data List	n*8		As defined by SPC-2 REQUEST SENSE CMD

SRP Flow Control Credit and Asynchronous Events IUs are depicted in the next table.

Field	Bits	Value	Notes
<i>SRP_CRED_REQ (16 bytes)</i>			
Type	8	81h	Target request to adjust Initiator <i>Request Limit</i>
Reserved	24		
Request Limit Delta	32		Initiator adds to Request Limit (flow control)
Tag	64		Uniquely identifies outstanding SRP request
<i>SRP_CRED_RSP (16 bytes)</i>			
Type	8	41h	Initiator response to SRP_CRED_REQ
Reserved	56		
Tag	64		Uniquely identifies outstanding SRP request
<i>SRP_AER_REQ (36 bytes plus length of sense data)</i>			
Type	8	82h	Target request to report an asynchronous event
Reserved	24		
Request Limit Delta	32		Initiator adds to Request Limit (flow control)
Tag	64		Uniquely identifies outstanding SRP request
Reserved	32		
Logical Unit Number	64		
Sense Data List Length	32	<i>n</i>	<i>n</i> is a multiple of 4
Reserved	32		
Sense Data List	<i>n*8</i>		As defined by SPC-2 REQUEST SENSE CMD
<i>SRP_AER_RSP (16 bytes)</i>			
Type	8	42h	Initiator response to SRP_AER_REQ
Reserved	56		
Tag	64		Uniquely identifies outstanding SRP request

SRP over InfiniBand:

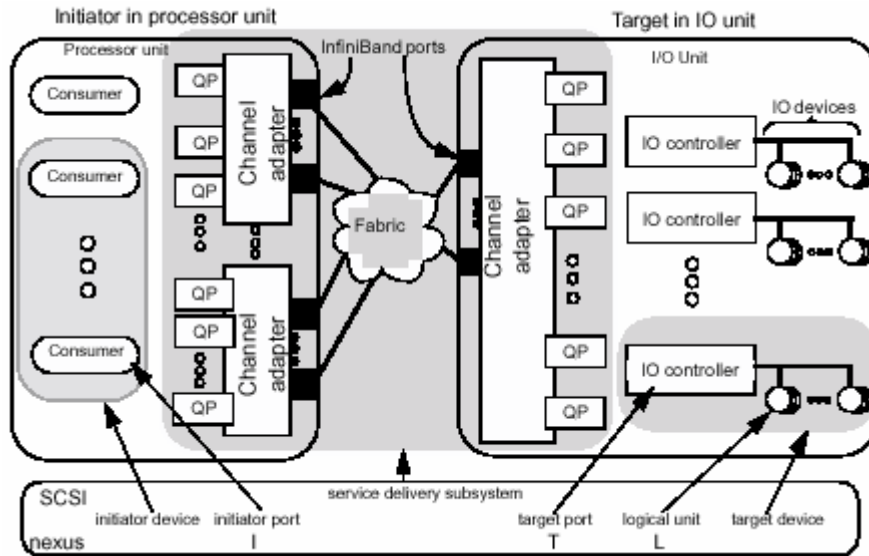
SRP initiator and target ports establish RC channels using an active-passive protocol. (Note that InfiniBand uses the terms "active-passive" for "client-server" and "active-active" for "peer-to-peer"). The initiator uses a CommMgt GMP to request each channel by specifying a Service ID associated with an IOC ServiceEntries table entry with a Service Name of "SRP.T10.NCITS" and including an SRP_LOGIN_REQ IU as Private Data. The target uses a CommMgt GMP to respond with a QPN and an SRP_LOGIN_RSP IU as Private Data. The initiator may then accept the connection with a CommMgt "Ready-to-Use" GMP or reject the connection by returning a "Reject" GMP with a Reason Code set to "Consumer Reject". A connection may be closed by either the initiator or target port by sending an SRP_LOGOUT IU. The receiver responds with an InfiniBand transport acknowledgement and disconnects.

The next two figures illustrate how the I_T_L nexus and SCSI initiator devices, initiator ports, target ports, and target devices map onto InfiniBand.

SRP targets include a device management agent to provide IOU, IOC, and ServiceEntries attributes and to provide a worldwide unique IOC GUID.

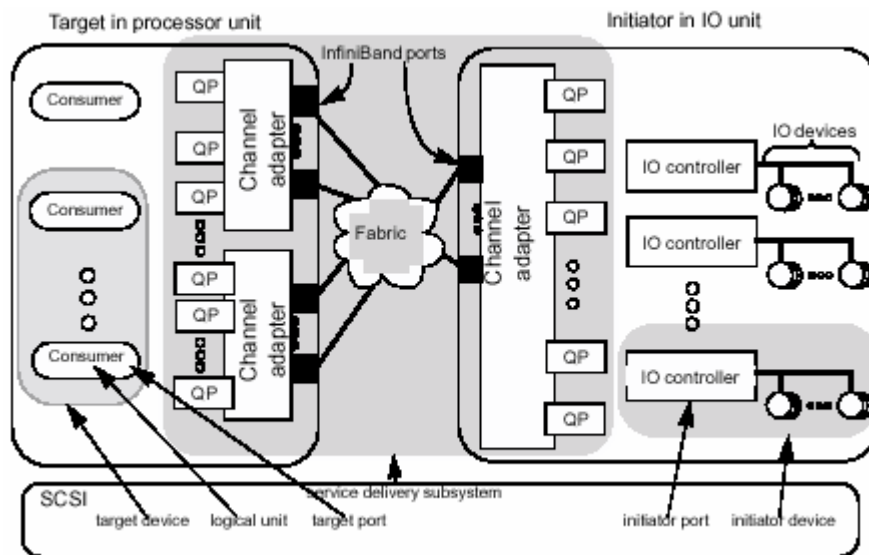
The next figure shows an initiator in a processor unit with a target in an I/O unit. The IOU must report an IOUnit attribute with Max Controllers and Controller List fields indicating at least one IOC. The IOC must report an IOControllerProfile attribute with SRP fields set appropriately and register a Service Name of "SRP.T10.NCITS".

- An SRP initiator device in a processor unit is one or more consumers.
- An SRP initiator port in a processor unit is a consumer.
- An SRP target port in an IOU is an IOC.
- An SRP target device in an IOU is an IOC plus one or more IO devices.



The next figure shows an initiator in an I/O unit and a target in a processor unit.

- An SRP initiator device in an IOU is an IOC.
- An SRP initiator port in an IOU is an IOC.
- An SRP target port in a processor unit is a consumer.
- An SRP target device in a processor unit is one or more consumers.
-



Terms related to SRP addressing are:

I_T nexus included in each IU specify a 128-bit initiator port identifier and a 128-bit target port identifier. An SRP port identifier is a worldwide unique identifier, constructed by concatenating a GUID with an 8-byte identifier extension.

Memory Descriptor is a 16-byte structure that identifies a memory segment, consisting of a 32-bit memory handle (start of the region in memory), a 64-bit

virtual address (start of segment in region), and a 32-bit length (of the segment). A target port may use a descriptor for an RDMA Read or an RDMA Write, but not both.

TransportID within the *Access Identifier* field in parameter data is used in SPC-3 Access Controls in a protocol and interconnect-specific manner. At any given time, an initiator may be associated with at most one TransportID and at most one AccessID. Multiple initiators may be associated with the same TransportID or AccessID. Its length is a multiple of four and at least 24 bytes long. Its first byte identifies the transport protocol:

- FCP (00h) uses the 64-bit port WWN to form a 24 byte TransportID.
- SPI (01h) uses a 16-bit SCSI address unique within a SCSI domain identified by a 32-bit relative port identifier to form a 24 byte TransportID.
- SBP (03h) uses a 64-bit node unique ID (EUI-64) for an initiator port to form a 24 byte TransportID.
- SRP (04h) uses a 128-bit initiator port identifier to form a 24 byte TransportID.
- iSCSI (05h) uses a iSCSI Name of an initiator node, up to 255 bytes in length, to form a TransportID of up to 260 bytes.