**Document Identifier: DSP0222**

**Date: 2023-06-02**

**Version: 1.2.0WIP99**

# Network Controller Sideband Interface (NC-SI) Specification

### Information for Work-in-Progress version:

**IMPORTANT:** This document is not a standard. It does not necessarily reflect the views of the DMTF or its members. Because this document is a Work in Progress, this document may still change, perhaps profoundly and without notice. This document is available for public review and comment until superseded.

**Provide any comments through the DMTF Feedback Portal:**
http://www.dmtf.org/standards/feedback

**Supersedes: 1.2.0WIP95**

**Document Class: Normative**

**Document Status: DMTF Work-in-Progress**

**Document Language: en-US**

This document's normative language is English. Translation into other languages is permitted.

# CONTENTS

# Figures

# Tables

# Foreword

The *Network Controller Sideband Interface (NC-SI) Specification* (DSP0222) was prepared by the PMCI Working Group.

This version supersedes version 1.2.0WIP95. For a list of changes, see the Change Log in ANNEX C.

DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability.

**Acknowledgments**

The DMTF acknowledges the following individuals for their contributions to this document:

**Editors:**

- Hemal Shah – Broadcom Inc.
- Bob Stevens – Dell Technologies

**Contributors:**

- Patrick Caporale – Lenovo
- Phil Chidester – Dell Inc.
- Yuval Itkin – NVIDIA Corporation
- Ira Kalman – Intel Corporation
- Patrick Kutch – Intel Corporation
- Eliel Louzoun – Intel Corporation
- Rob Mapes – Marvell Corporation
- Edward Newman – Hewlett Packard Enterprise
- Patrick Schoeller – Intel Corporation
- Tom Slaight – Intel Corporation
- Dov Goldstein – Intel Corporation
- Jason Kilpatrick – Dell Technologies

# Introduction

In out-of-band management environments, the interface between the out-of-band Management Controller and the Network Controller is critical. This interface is responsible for supporting communication between the Management Controller and external management applications.

The goal of this specification is to define an interoperable sideband communication interface standard to enable the exchange of management data between the Management Controller and Network Controller. The Sideband Interface is intended to provide network access for the Management Controller, and the Management Controller is expected to perform all the required network functions.

This specification defines the protocol and commands necessary for the operation of the sideband communication interface. This specification also defines physical and electrical characteristics of a sideband binding interface that is a variant of RMII targeted specifically for sideband communication traffic.

The specification is primarily intended for architects and engineers involved in the development of Network and Management Controllers that will be used in providing out-of-band management functionality.

# Network Controller Sideband Interface (NC-SI) Specification

## 1 Scope

This specification defines the functionality and behavior of the Sideband Interface responsible for connecting the Network Controller (including Ethernet, Fibre Channel, and InfiniBand controllers) to the Management Controller. It also outlines the behavioral model of the network traffic destined for the Management Controller from the Network Controller.

This specification defines the following two aspects of the Network Controller Sideband Interface (NC-SI):

- behavior of the interface, which include its operational states as well as the states of the associated components
- the payloads and commands of the communication protocol supported over the interface

The scope of this specification is limited to addressing only a single Management Controller communicating with one or more Network Controllers.

This specification also defines the following aspects of a 3.3V RMII-Based Transport (RBT) based physical medium:

- transport binding for NC-SI over RBT
- electrical and timing requirements for the RBT
- an optional hardware arbitration mechanism for RBT

Only the topics that may affect the behavior of the Network Controller or Management Controller, as it pertains to the Sideband Interface operations, are discussed in this specification.

## 2 Normative references

The following referenced documents are indispensable for the application of this document. For dated or versioned references, only the edition cited (including any corrigenda or DMTF update versions) applies. For references without a date or version, the latest published edition of the referenced document (including any corrigenda or DMTF update versions) applies.

DMTF DSP0240, *Platform Level Data Model (PLDM) Base Specification* 1.0
https://www.dmtf.org/dsp/DSP0240

DMTF DSP0261, *NC-SI over MCTP Binding Specification* 1.2
https://www.dmtf.org/dsp/DSP0261

DMTF DSP0274, *Security Protocol and Data Model (SPDM) Specification* 1.1 & 1.2
https://www.dmtf.org/dsp/DSP0274

IEEE 802.3*, IEEE Standard for Ethernet,* June 2018
https://standards.ieee.org/ieee/802.3/7071/

IETF, RFC4122, *A Universally Unique Identifier (UUID) URN Namespace*, July 2005
http://datatracker.ietf.org/doc/rfc4122/

InfiniBand™ Architecture Specification
https://www.infinibandta.org/ibta-specification/

ISO/IEC Directives, Part 2, *Principles and rules for the structure and drafting of ISO and IEC documents*
http://isotc.iso.org/livelink/livelink?func=ll&objId=4230456&objAction=browse&sort=subtype

Reduced Media Independent Interface (RMII) Consortium, *RMII Specification*, revision 1.2, March 20, 1998
http://ebook.pldworld.com/_eBook/-Telecommunications,Networks-/TCPIP/RMII/rmii_rev12.pdf

CMIS, Common Management Interface Specification 4.0 / 5.0 / 5.1
https://www.oiforum.com/documents/archived-non-oif-generated-specifications/

CMIS, Common Management Interface Specification 5.2
https://www.oiforum.com/wp-content/uploads/OIF-CMIS-05.2.pdf

SFF, SFF-8024, SFF Cross Reference to Industry Products
https://www.snia.org/technology-communities/sff/specifications

SFF, SFF-8436, QSFP+ 10Gbs 4X Pluggable Transceiver
https://www.snia.org/technology-communities/sff/specifications

SFF, SFF-8472, Diagnostic Monitoring Interface for Optical Transceivers
https://www.snia.org/technology-communities/sff/specifications

SFF, SFF-8636, Management Interface for Cabled Environments
https://www.snia.org/technology-communities/sff/specifications

DSFP, Dual Small Form Factor Pluggable Module, 1.0
https://dsfpmsa.org/wp-content/uploads/2021/07/DSFP_Module_Specification.pdf

Fibre Channel Technical Committee (ANSI/INCITS TC T11)
http://www.t11.org and http://www.incits.org

# 3 Terms and definitions

## 3.1 Wording Interpretation

In this document, some terms have a specific meaning beyond the normal English meaning. Those terms are defined in this clause.

The terms "shall" ("required"), "shall not", "should" ("recommended"), "should not" ("not recommended"), "may", "need not" ("not required"), and "can" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, Clause 7. The terms in parentheses are alternatives for the preceding term, for use in exceptional cases when the preceding term cannot be used for linguistic reasons. Note that ISO/IEC Directives, Part 2, Clause 7 specifies additional alternatives. Occurrences of such additional alternatives shall be interpreted in their normal English meaning.

The terms "clause", "subclause", "paragraph", and "annex" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, Clause 6.

The terms "normative" and "informative" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, Clause 3. In this document, clauses, subclauses, or annexes labeled "(informative)" do not contain normative content. Notes and examples are always informative elements.

The terms defined in DSP0004, DSP0223, and DSP1001 apply to this document. The following additional terms are used in this document.

## 3.2 Requirement term definitions

This clause defines key phrases and words that denote requirement levels in this specification.

**3.2.1**

**can**

indicates an ability or capability expressed by the specification or of the possibility of some outcome in the context of the specification

**3.2.2**

**cannot**

indicates the inability or denial of the possibility of a certain outcome in the context of the specification

**3.2.3**

**conditional**

indicates that an item is required under specified conditions

**3.2.4**

**deprecated**

indicates that an element or profile behavior has been outdated by newer constructs

**3.2.5**

**mandatory**

indicates that an item is required under all conditions

**3.2.6**

**may**

a permission expressed by this specification

**3.2.7**

**may not**

an expression of permission in the negative; a lack of requirement

**3.2.8**

**not recommended**

indicates that valid reasons may exist in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and carefully weighed before implementing any behavior described with this label

**3.2.9**

**obsolete**

indicates that an item was defined in prior specifications but has been removed from this specification

**3.2.10**

**optional**

indicates that an item is not mandatory, conditional, or prohibited

**3.2.11**

**recommended**

indicates that valid reasons may exist in particular circumstances to ignore a particular item, but the full implications should be understood and carefully weighed before choosing a different course

**3.2.12**

**required**

indicates that the item is an absolute requirement of the specification

**3.2.13**

**shall**

indicates that the item is an absolute requirement of the specification

**3.2.14**

**shall not**

indicates that the item is an absolute prohibition of the specification

**3.2.15**

**should**

indicates a recommendation of the specification, but the full implications should be understood and carefully weighed before choosing a different course

**3.2.16**

**should not**

indicates a recommendation against, but the full implications should be understood and carefully weighed before implementing any behavior described with this label

## 3.3 NC-SI term definitions

For the purposes of this document, the following terms and definitions apply.

**3.3.1**

**frame**

a data packet of fixed or variable length that has been encoded for digital transmission over a node-to-node link

*Frame* is used in references to IEEE 802.3 Frames. *Packet* is used in all other references.

**3.3.2**

**packet**

a formatted block of information carried by a computer network

*Frame* is used in references to IEEE 802.3 Frames. *Packet* is used in all other references.

**3.3.3**

**external network interface**

the interface of the Network Controller that provides connectivity to the external network infrastructure; also known as *port*

**3.3.4**

**internal host interface**

the interface of the Network Controller that provides connectivity to the host operating system running on the platform

### 3.3.5
### Management Controller

an intelligent entity composed of hardware/firmware/software that resides within a platform and is responsible for some or all of the management functions associated with the platform; also known as BMC and Service Processor

### 3.3.6
### Network Controller

the component within a system that is responsible for providing connectivity to an external Ethernet, Fibre Channel, or InfiniBand network

### 3.3.7
### remote media

a manageability feature that enables remote media devices to appear as if they are attached locally to the host

### 3.3.8
### Network Controller Sideband Interface
### NC-SI

The RBT interface of the Network Controller that provides network connectivity to a Management Controller; also shown as *Sideband Interface, RBT* or *NC-SI* as appropriate in the context

### 3.3.9
### integrated controller

a Network Controller device that supports two or more channels for the NC-SI that share a common NC-SI physical interface (for example, a Network Controller that has two or more physical network ports and a single NC-SI bus connection)

### 3.3.10
### multi-drop

refers to the situation in which multiple physical communication devices share an electrically common bus and a single device acts as the master of the bus and communicates with multiple “slave” or “target” devices

Related to NC-SI, a Management Controller serves the role of the master, and the Network Controllers are the target devices

### 3.3.11
### point-to-point

refers to the situation in which only a single Management Controller and single Network Controller package are used on the bus in a master/slave relationship, where the Management Controller is the master

### 3.3.12
### Channel

refers to the logical representation of a network port in a Network Controller that supports Control traffic and may support Pass-through traffic

A Network Controller may have a 1:1 relationship of NC-SI channels to physical network ports, or Network Controllers that support partitioning can have multiple channels on a given network port

### 3.3.13
### Partition

one or more NC-SI channels in a Network Controller that share a common network port

### 3.3.14
### Package

one or more NC-SI channels in a Network Controller that share a common set of electrical buffers and common electrical buffer controls for the NC-SI bus

Typically a single, logical NC-SI package exists for a single physical Network Controller package (chip or module). However, this specification allows a single physical chip or module to hold multiple NC-SI logical packages

### 3.3.15
### control traffic
### Control Packets
### control packets

command, response, and asynchronous event notification packets transmitted between the Management Controller and Network Controllers for the purpose of managing the NC and NC-SI

### 3.3.16
### Command

Control Packet sent by the Management Controller to the Network Controller to request the Network Controller to perform an action, and/or return data

### 3.3.17
### Response

Control Packet sent by the Network Controller to the Management Controller as a positive acknowledgement of a command received from the Management Controller, and to provide the execution outcome of the command, as well as to return any required data

### 3.3.18
### Asynchronous Event Notification

Control Packet sent by the Network Controller to the Management Controller as an explicit notification of the occurrence of an event of interest to the Management Controller

### 3.3.19
### pass-through traffic
### pass-through packets

network packets passed between the external network and the Management Controller through the Network Controller

### 3.3.20
### RBT
### RMII-Based Transport

Electrical and timing specification for a 3.3V-signaling physical medium that is derived from RMII

### 3.3.21
### PCIe Endpoint

Also PCIe Port, physically the collection of Transmitters and Receivers located on the same chip that define a Link, logically the interface between a component and a PCI Express Link. For the purposes of this specification, it is a PCIe upstream port on the NC that is assigned a PCI Bus number when connecting to a PCIe Switch or Root Complex

**3.3.22**

**PCIe Link**

The collection of two Ports and their interconnecting Lanes. A Link is a dual-simplex communications path between two components.

## 3.4 Numbers and number bases

Numbers in this specification are written as follows:

- Hexadecimal numbers are written with a "`0x`" prefix (for example, `0xFF` and `0x80`).
- Binary numbers are written with a lowercase "`b`" suffix (for example, `1001b` and `10b`).
- Hexadecimal and binary numbers are formatted in the Courier New font.
- "uint8" describes an unsigned 8-bit integer value.

## 3.5 Network Addresses

Network addresses in this specification are written as follows:

- IPv4 addresses are written as decimal numbers with period (.) separators
- IPv6 addresses are written as hexadecimal numbers with colon (:) separators
- MAC addresses are written as 6 hexadecimal number pairs with colon (:) separators
- InfiniBand GUIDs are written as hexadecimal numbers with no separators
- Fibre Channel WWNs are written as hexadecimal numbers with no separators

## 3.6 Reserved fields

Unless otherwise specified, reserved fields (bytes, bits, etc.) are reserved for future use and should be written as zeros and ignored when read.

# 4 Acronyms and abbreviations

The following symbols and abbreviations are used in this document.

**4.1**

**AC**

Alternating Current

**4.2**

**AEN**

Asynchronous Event Notification

**4.3**

**BMC**

Baseboard Management Controller (often used interchangeably with MC)

**4.4**

**CMIS**

Common Management Interface Specification

**4.5**
**CRC**

Cyclic Redundancy Check

**4.6**
**CRS_DV**

a physical NC-SI signal used to indicate Carrier Sense/Received Data Valid

**4.7**
**DC**

Direct Current

**4.8**
**DHCP**

Dynamic Host Configuration Protocol

**4.9**
**EEE**

Energy Efficient Ethernet

**4.10**
**FC**
**Fibre Channel**

**4.11**
**FCS**

Frame Check Sequence

**4.12**
**IB**

InfiniBand

**4.13**
**MC**

Management Controller

**4.14**
**NC**

Network Controller

**4.15**
**NC-SI**

Network Controller Sideband Interface

**4.16**
**NC-SI RX**

the direction of traffic on RBT from the Network Controller to the Management Controller

**4.17**
**NC-SI TX**

the direction of traffic RBT to the Network Controller from the Management Controller

**4.18**

**RMII**

Reduced Media Independent Interface

**4.19**

**RX**

Receive

**4.20**

**RXD**

physical NC-SI signals used to transmit data from the Network Controller to the Management Controller

**4.21**

**RX_ER**

a physical NC-SI signal used to indicate a Receive Error

**4.22**

**SerDes**

serializer/deserializer; an integrated circuit (IC or chip) transceiver that converts parallel data to serial data and vice-versa. This is used to support interfaces such as 1000Base-X and others.

**4.23**

**SFF**

Small Form Factor

**4.24**

**TX**

Transmit

**4.25**

**TXD**

physical NC-SI signals used to transmit data from the Management Controller to the Network Controller

**4.26**

**VLAN**

Virtual LAN

# 5 NC-SI overview

## 5.1 General

This specification enables a common interface definition between different Management Controller and Network Controller vendors. This specification addresses not only the electrical and protocol specifications, but also the system-level behaviors for the Network Controller and the Management Controller related to the NC-SI.

The NC-SI is defined as the interface (protocol, messages, and medium) between a Management Controller and one or more Network Controllers. This interface, referred to as a Sideband Interface in Figure 1, is responsible for providing external network connectivity for the Management Controller while also allowing the external network interface to be shared with traffic to and from the host.

The specification of how the NC-SI protocol and messages are implemented over a particular physical medium is referred to as a transport binding. This document, DSP0222, includes the definition of the transport binding, electrical, framing, and timing specifications for a physical interface called RBT (RMII-based Transport). Electrically, RBT, as described in clause 10, is similar to the Reduced Media Independent Interface™ (RMII) – see ANNEX B. Transport bindings for NC-SI over other media and transport protocols are defined through external transport binding specifications, such as DSP0261, the *NC-SI over MCTP Transport Binding Specification*. That specification defines the Get Supported Media command (`0x54`) which is used to discover support for operations over multiple media types. This command may be issued on any NC-SI transport including RBT.

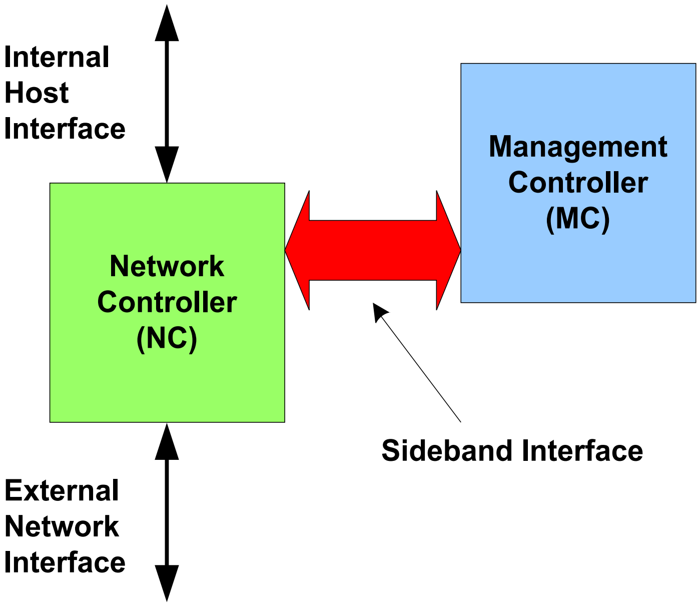


**Figure 1 – NC-SI functional block diagram**

NC-SI traffic flow is illustrated in Figure 2. Two classes of packet data can be delivered over the Sideband Interface:

- “Pass-through” packets that are transferred between the Management Controller and the external network and/or an internal host.
- “Control” packets that are transferred between the Management Controller and Network Controllers for control or configuration functionality. This specification defines NC-SI commands and responses as well as a mechanism to customize and extend functionality via OEM commands extensions – see ANNEX A.

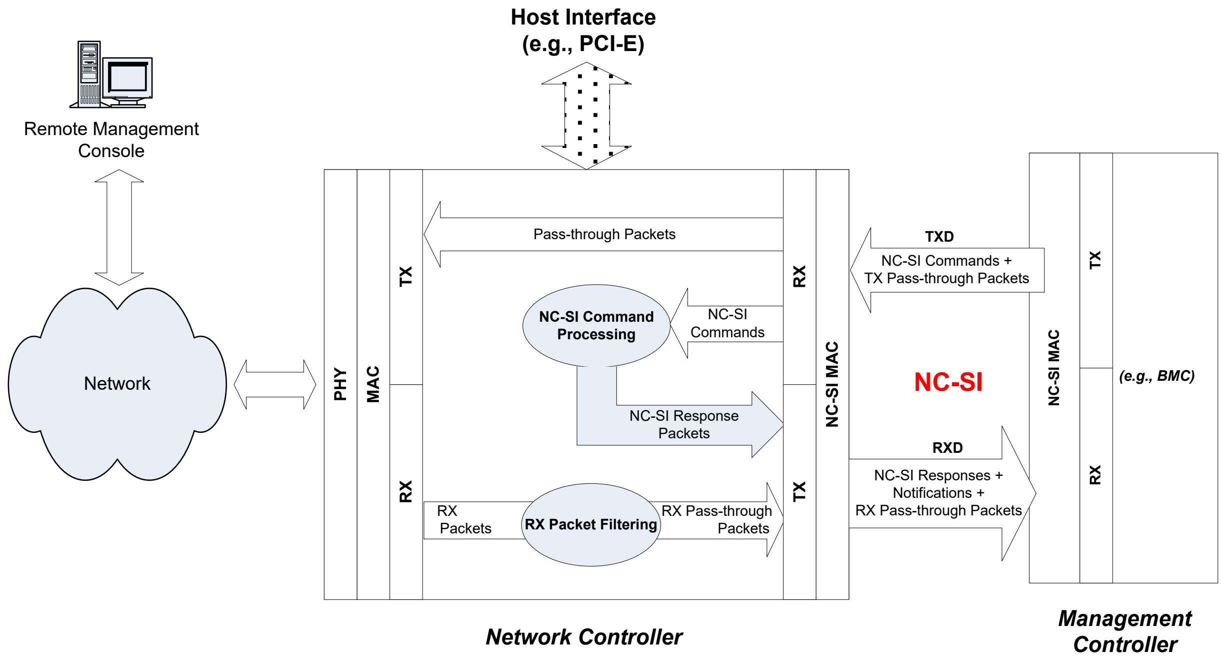


**Figure 2 – NC-SI RBT traffic flow diagram**

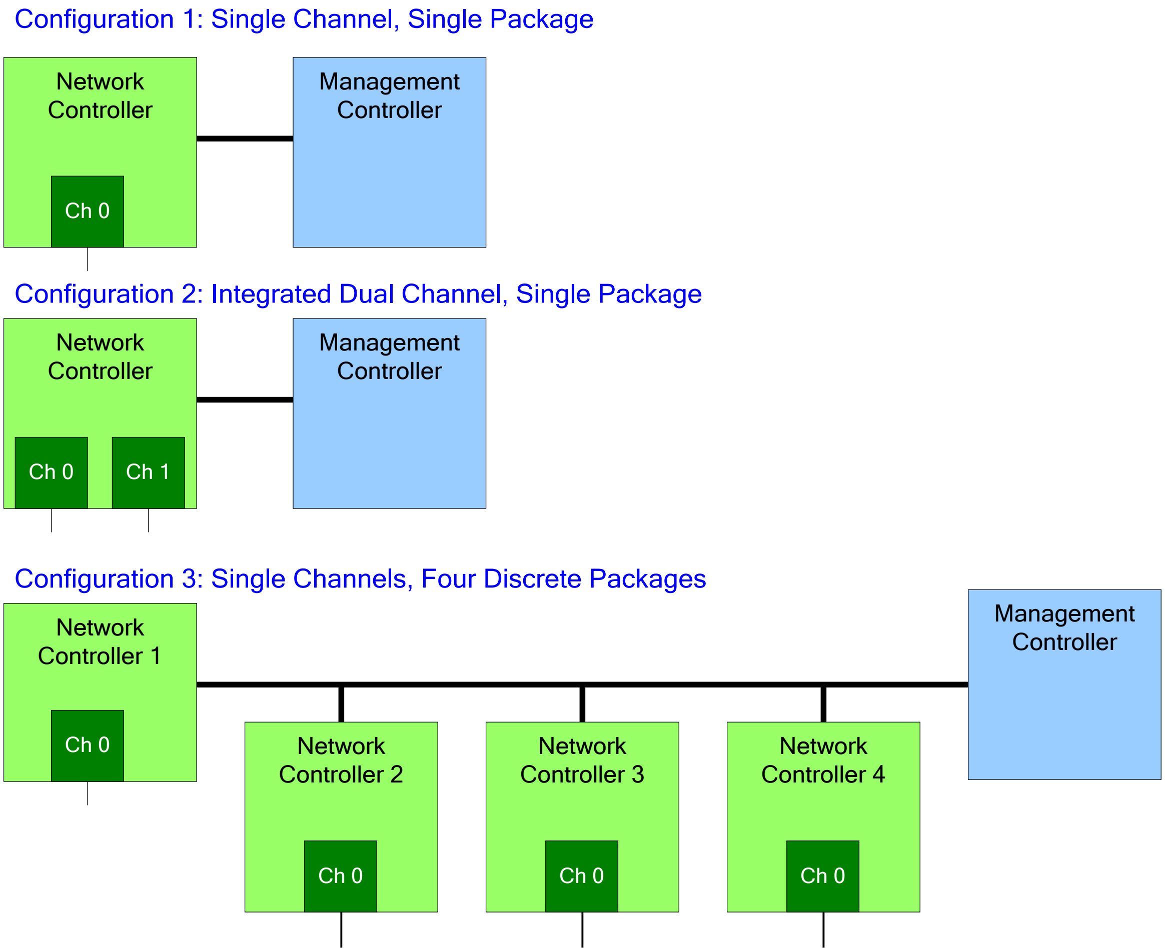
NC-SI is intended to operate independently from the in-band activities of the Network Controller. As such, the Sideband Interface is not specified to be visible through the host interface of the Network Controller. From the external world, this interface should behave and operate like a standard Ethernet Interface.

## 5.2 Defined topologies

The topologies supported under this specification apply to the case in which a single Management Controller is actively communicating with one or more Network Controllers on the Sideband Interface over RBT. The RBT electrical specification is targeted to directly support up to four physical Network Controller packages. The protocol specification allows up to eight Network Controller packages, with up to 31 channels per package.

Figure 3 illustrates some examples of Network Controller configurations supported by the NC-SI in the current release:

- Configuration 1 shows a Management Controller connecting to a single Network Controller with a single external network connection.
- Configuration 2 shows a Management Controller connecting to a Network Controller package that supports two NC-SI channel connections.
- Configuration 3 shows a Management Controller connecting to four discrete Network Controllers.



Configuration 1: Single Channel, Single Package

Configuration 2: Integrated Dual Channel, Single Package

Configuration 3: Single Channels, Four Discrete Packages

**Figure 3 – Example topologies supported by the NC-SI**

## 5.3 Single and integrated Network Controller implementations

This clause illustrates the general relationship between channels, packages, receive buffers, and bus buffers for different controller implementations.

An integrated controller is a Network Controller that connects to the NC-SI RBT (or other physical interfaces that support NC-SI) interface and provides NC-SI support for two or more network connections. A single controller is a controller that supports only a single NC-SI channel.

For the *NC-SI Specification*, an integrated controller can be logically implemented in one of three basic ways, as illustrated in Figure 4. Although only two channels are shown in the illustration, an integrated controller implementation can provide more than two channels. The example channel and package numbers (for example, channel 0, package 0) refer to the Internal Channel and Package ID subfields of the Channel ID. For more information, see clause 6.1.9.

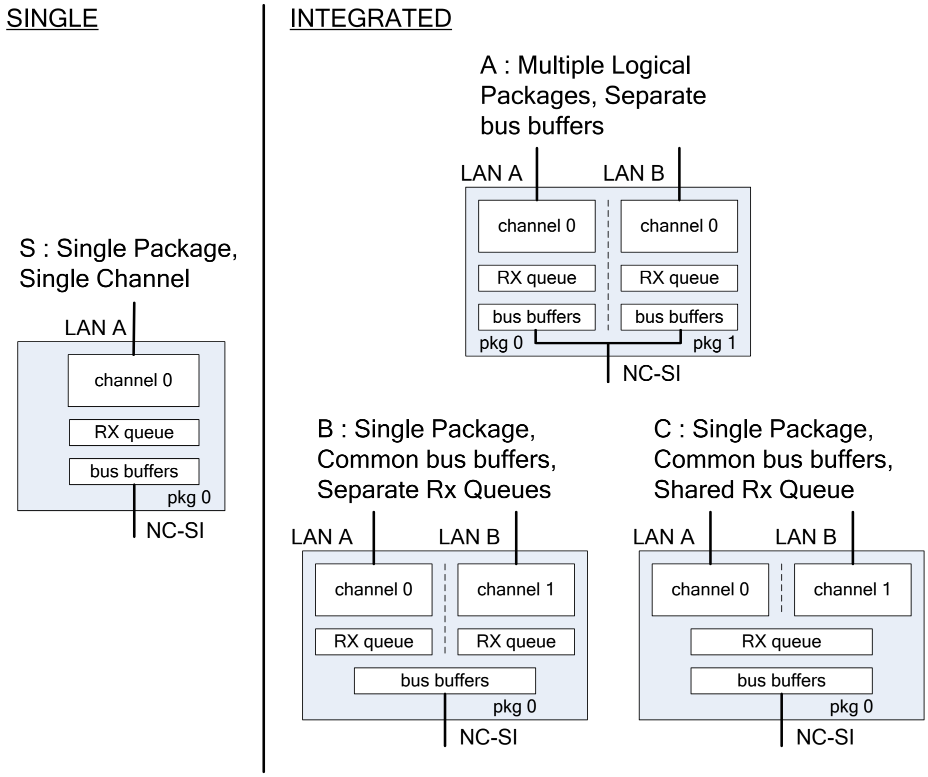


**Figure 4 – Network Controller integration options**

Packages that include multiple channels are required to handle internal arbitration between those channels and the Sideband Interface. The mechanism by which this occurs is vendor-specific and not specified in this document. This internal arbitration is always active by default. No NC-SI commands are defined for enabling or disabling internal arbitration between channels.

The following classifications refer to a logical definition. The different implementations are distinguished by their behavior with respect to the NC-SI bus and command operation. The actual physical and internal implementation can vary from the simple diagrams. For example, an implementation can act as if it has separate RX queues without having physically separated memory blocks for implementing those queues.

- **S: Single Package, Single Channel**

  This implementation has a single NC-SI interface providing NC-SI support for a single LAN port, all contained within a package or module that has a single connection to the NC-SI physical bus. Note that FC Bonding is supported in this specification and thus multiple physical ports may be aggregated into one logical port.

- **A: Multiple Logical Packages, Separate Bus Buffers**

  This implementation acts like two physically separate Network Controllers that happen to share a common overall physical container. Electrically, they behave as if they have separate electrical buffers connecting to the NC-SI bus. This behavior might be accomplished by means of a passive internal bus or by separate physical pins coming from the overall package. From the point of view of the Management Controller and the NC-SI command operation, this implementation behaves as if the logical controllers were implemented as physically separate controllers.

  This type of implementation could include internal hardware arbitration between the two logical Network Controller packages. If hardware arbitration is provided external to the package, it shall meet the requirements for hardware arbitration described later in this specification. (For more information, see clause 7.3.)

- **B: Single Package, Common Bus Buffers, Separate RX Queues**

  In this implementation, the two internal NC-SI channels share a common set of electrical bus buffers. A single Deselect Package command will deselect the entire package. The Channel Enable and Channel Disable commands to each channel control whether the channel can transmit Pass-through and AEN packets through the NC-SI interface. The Channel Enable command also determines whether the packets to be transmitted through the NC-SI interface will be queued up in an RX Queue for the channel while the channel is disabled or while the package is deselected. Because each channel has its own RX Queue, this queuing can be configured for each channel independently.

- **C: Single Package, Common Bus Buffers, Shared RX Queue**

  This implementation is the same as described in the preceding implementation, except that the channels share a common RX Queue for holding Pass-through packets to be transmitted through the NC-SI interface. This queue could also queue up AEN or Response packets.

In addition to the general purpose architectures listed above, some Network Controllers support more advanced architectures that provide for multiple host interfaces that share a single channel/physical port (commonly called partitions), a single host interface that sends and receives traffic over multiple physical ports, but modeled as a single channel, and lastly an internally terminated channel that can be used to control some other functionality in the NC that requires a communication and control path to the MC.

## 5.4 Transport stack

The overall transport stack of the NC-SI is illustrated in Figure 5. The lowest level is the physical-level interface (for example, RBT), and the media-level interface is based on Ethernet. Above these interfaces are the two data-level protocols that are supported by the *NC-SI Specification*: NC-SI Command Protocol and the Network Data Protocol (for example, ARP, IP, DHCP, and NetBIOS) associated with Pass-through traffic for NCs. Both protocols are independent from binding to the underlying physical interface. This specification only defines the binding for NC-SI over RBT.

This document defines the necessary NC-SI command set and interface specification that allows the appropriate configuration of the Network Controller parameters and operation to enable network traffic to flow to and from external networks to the Management Controller for those devices that support it. As shown in Figure 5, the scope of the NC-SI Command Protocol is limited to the interface between the Network Controller and the Management Controller.

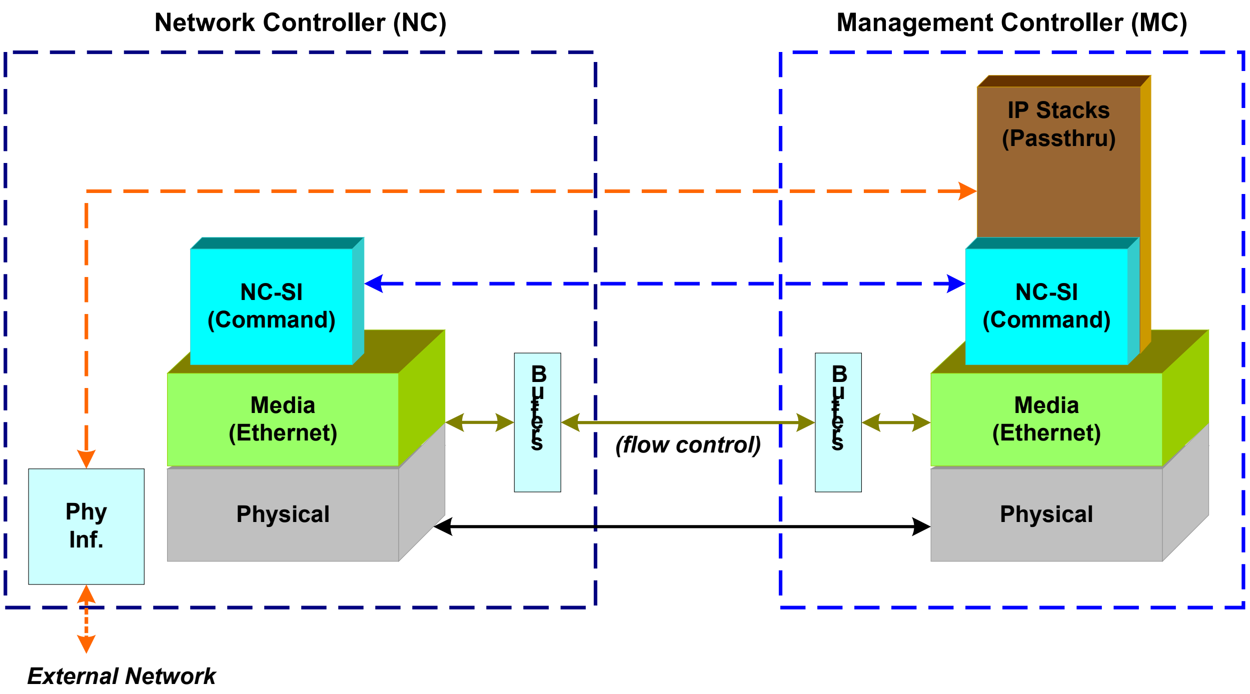


**Figure 5 – NC-SI transport stack**

## 5.5 Transport protocol

A simple transport protocol is used to track the reliable reception of command packets. The transport protocol is based upon a command/response paradigm and involves the use of unique Instance IDs (IIDs) in the packet headers to allow responses received to be matched to previously transmitted commands. The Management Controller is the generator of command packets sent to the Sideband Interface of one or more Network Controllers in the system, and it receives response packets from them. A response packet is expected to be received for every command packet successfully sent.

The transport protocol described here shall apply only to command and response packets sent between the Management Controller and the Network Controller.

## 5.6 Byte and bit ordering for transmission

Unless otherwise specified, the bytes for a multi-byte numeric field are transmitted most significant byte first and bits within a byte are transmitted most significant bit first.

# 6 Operational behaviors

## 6.1 Typical operational model

This clause describes the typical system-level operation of the NC-SI components.

The following tasks are associated with Management Controller use of the NC-SI:

- **Initial configuration**

  When the NC-SI interface is first powered up, the Management Controller needs to discover and configure NC-SI devices as well as to enable pass-through operation. This task includes setting parameters such as MAC addresses, configuring Layer 2 filtering, setting Channel enables, and so on.

- **General Controller configuration and monitoring**

  The Management Controller may also configure and monitor aspects of Controller operation.

- **Pass-through**

  The Management Controller handles transmitting and receiving Pass-through packets using the NC-SI. Pass-through packets can be delivered to and received from the network through the NC-SI based on the Network Controller's NC-SI configuration.

- **Asynchronous event handling**

  In certain situations, a status change in the Network Controller, such as a Link State change, can generate an asynchronous event on the Sideband Interface. These event notifications are sent to the Management Controller where they are processed as appropriate.

- **Error handling**

  The Management Controller handles errors that could occur during operation or configuration. For example, a Network Controller might have an internal state change that causes it to enter a state in which it requires a level of reconfiguration (this condition is called the "Initial State," described in more detail in 6.1.4); or a data glitch on the NC-SI could have caused an NC-SI command to be dropped by the Network Controller, requiring the Management Controller to retry the command.

### 6.1.1 State definitions and defined states

Table 1 describes states related to whether and when the Network Controller is ready to handle NC-SI command packets, when it is allowed to transmit packets through the NC-SI interface, and when it has entered a state where it is expecting configuration by the Management Controller.

**Table 1 – NC-SI operating state descriptions**

| State | Applies to | Description |
|---|---|---|
| Interface Power Down | Package | The NC-SI is in the power down state. |
| Interface Power Up | Package | The NC-SI is in the power up state, as defined in clause 10. |
| Package Selected (also referred to as the Selected state) | Package | A Selected package is allowed to turn on its electrical buffers and transmit through the NC-SI interface. |
| Package Deselected (also referred to as the Deselected state) | Package | A Deselected package is not allowed to turn on its electrical buffers and transmit through the NC-SI interface. |

| State | Applies to | Description |
|---|---|---|
| Hardware Arbitration Enabled | Package | When hardware arbitration is enabled, the package is allowed to transmit through the NC-SI interface only when it is Selected and has the TOKEN opcode. |
| Hardware Arbitration Disabled | Package | When hardware arbitration is disabled, the package is allowed to transmit through the NC-SI interface anytime that it is Selected, regardless of whether it has the TOKEN opcode. |
| Package Ready | Package | In the Package Ready state, the package is able to accept and respond to NC-SI commands for the package and be Selected. |
| Package Not Ready | Package | The Package Not Ready state is a transient state in which the package does not accept package-specific commands. |
| Channel Ready | Channel | In the Channel Ready state, a channel within the package is able to accept channel-specific NC-SI commands that are addressed to its Channel ID (Package ID + Internal Channel ID). |
| Channel Not Ready | Channel | The Channel Not Ready state is a transient state in which the channel does not accept channel-specific commands. |
| Initial State | Channel | In the Initial State, the channel is able to accept and respond to NC-SI commands, and one or more configuration settings for the channel need to be set or restored by the Management Controller (that is, the channel has not yet been initialized, or has encountered a condition where one or more settings have been lost and shall be restored). Refer to 6.1.4 for more information. |
| Channel Enabled | Channel | This is a sub-state of the Channel Ready state. When a channel is enabled, the channel is allowed to transmit unrequested packets (that is, packets that are not command responses — for example, AEN and Pass-through packets) through the NC-SI interface whenever the package is Selected. |
| Channel Disabled | Channel | This is a sub-state of the Channel Ready state. When a channel is disabled, the channel is not allowed to transmit unrequested packets (that is, packets that are not command responses — for example, AEN and Pass-through packets) through the NC-SI interface. |

### 6.1.2 NC-SI RBT pre-operational states

There are two states defined on RBT before it becomes operational:

- NC-SI Interface Power Down state

  In this state, the NC-SI Physical interface and the associated receive and transmit buffers in all devices on the NC-SI RBT (that is, the NC-SI interfaces on the Network Controllers and Management Controller) are not powered up.

- NC-SI Power Up state

  In this state, the NC-SI Physical interface and the associated receive and transmit buffers in all devices on the NC-SI RBT (that is, the Network Controller and Management Controller) are powered up.

  NOTE: NC transmit I/O buffers should not be enabled in this state. The Network Controller is expected to transition to the Initial State within T4 seconds after the Power Up state is entered.

### 6.1.3 Package Ready state

A Network Controller in the Package Ready state shall be able to respond to any NC-SI commands that are directed to the ID for the overall package (versus being directed to a particular channel within the package). Package-specific commands are identified by a particular set of Channel ID values delivered in the command header (see clause 6.1.9).

### 6.1.4 Initial State

The Initial State for a channel corresponds to a condition in which the Sideband Interface is powered up and is able to accept NC-SI commands, and the channel has one or more configuration settings that need to be set or restored by the Management Controller. Unless default configuration settings are explicitly defined in this specification, the default values are implementation specific. The MC should not make any assumptions on any configuration settings that are not defined in this specification. Because this state may be entered at any time, the Initial State shall be acknowledged with a Clear Initial State command for the Initial State to be exited. This requirement helps to ensure that the Management Controller does not continue operating the interface unaware that the NC-SI configuration had autonomously changed in the Network Controller.

An NC-SI channel in the Initial State shall:

- be able to respond to NC-SI commands that are directed to the Channel ID for the particular channel (see clause 6.1.9)
- respond to all non-OEM NC-SI command packets that are directed to the channel or partitions on the channel with a Response Packet that contains a Response Code of “Command Failed” and a Reason Code of “Initialization Required”
- place the channel into the Disabled state
- set hardware arbitration (if supported) to “enabled” on Interface Power Up only; otherwise, the setting that was in effect before entry into the Initial State shall be preserved (that is, the hardware arbitration enable/disable configuration is preserved across entries into the Initial State)
- set the enabled/disabled settings for the individual MAC and VLAN filters (typically set using the Set MAC Address, Set VLAN Filter, and Enable VLAN commands) to “disabled”

  NOTE It is recommended that global multicast and broadcast filters are also set to “disabled”.
- reset all counters defined in the various channel and partition level statistics commands, and the Get NC-SI Pass-Through Statistics command to `0x0`
- disable the Channel Network TX setting and transmission of Pass-through packets onto the network
- clear any record of prior command instances received upon entry into the Initial State (that is, assume that the first command received after entering the Initial State is a new command and not a retried command, regardless of any Instance ID that it may have received before entering the Initial State)
- disable transmission of AENs and reset any enabled AENs

Otherwise, there is no requirement that other NC-SI configuration settings be set, retained, or restored to particular values in the Initial State unless otherwise specified. Controller configuration settings that are identified as persistent and saved to NVRAM are one example of retained settings.

The Initial State is a NC-SI configuration state and therefore places no requirements on the NC’s network link state.

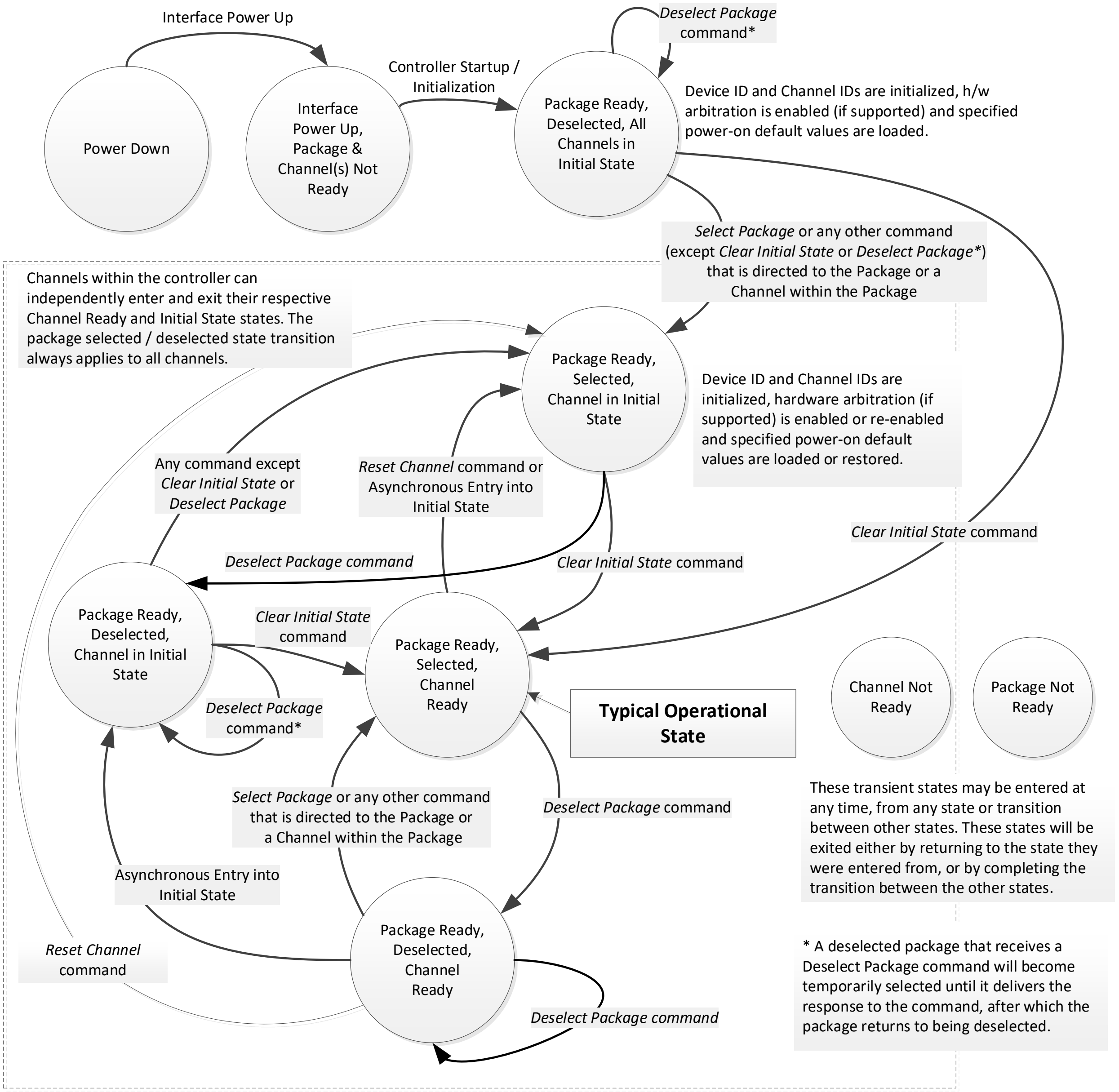
### 6.1.5 NC-SI Initial State recovery

As described in clause 6.1.4, a channel in the Initial State shall receive the Clear Initial State command before other commands can be executed. This requirement ensures that if the Initial State is entered asynchronously, the Management Controller is made aware that one or more NC-SI settings may have changed without its involvement and blocks the Management Controller from issuing additional commands under that condition. Until the channel receives the Clear Initial State command, the Network Controller shall respond to any other received command directed to the channel or partitions on the channel with a Command Failed response code and Interface Initialization Required reason code to indicate that the Clear Initial State command shall be sent. See response and reason code definitions in clause 8.2.5.2.

If the Management Controller, at any time, receives the response indicating that the Clear Initial State command is expected, it should interpret this response to mean that default settings have been restored for the channel (per the Initial State specification), and that one or more package/channel settings need to be restored by the Management Controller.

### 6.1.6 State transition diagram

Figure 6 illustrates the general relationship between the package- and channel-related states described in Table 1 and the actions that cause transitions between the states. Each bubble in Figure 6 represents a particular combination of states as defined in Table 1.

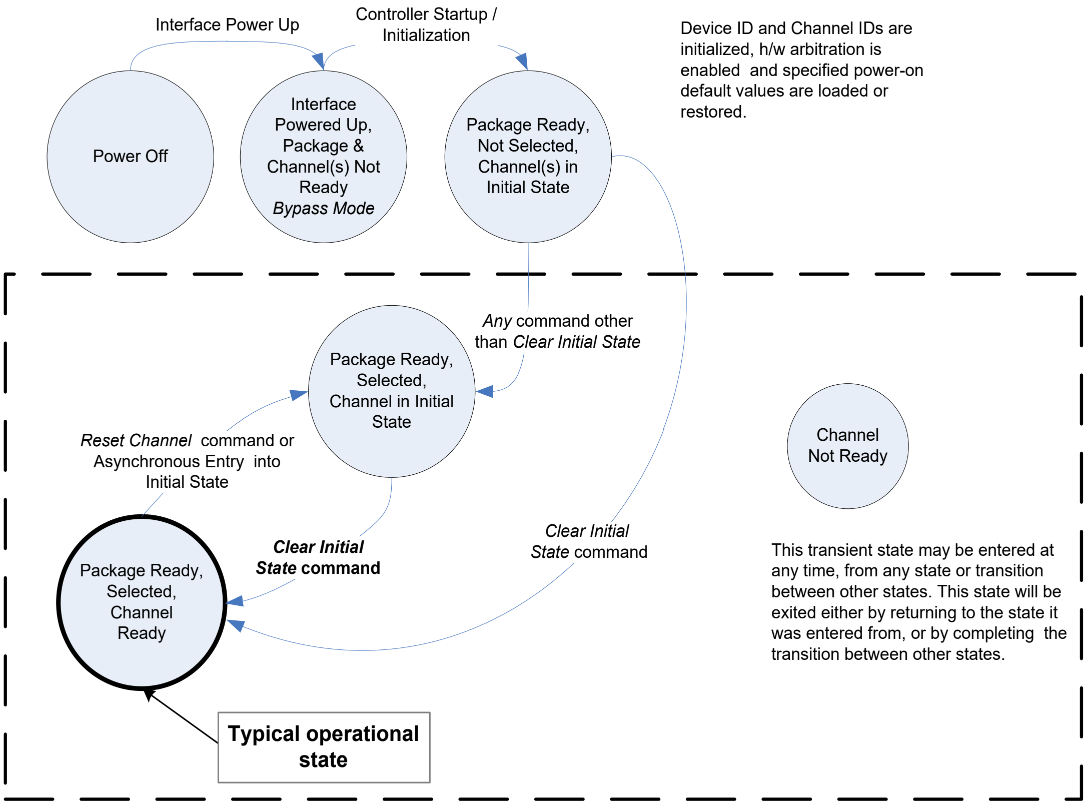

Interface Power Up
Power Down
Interface Power Up, Package & Channel(s) Not Ready
Controller Startup / Initialization
Deselect Package command*
Package Ready, Deselected, All Channels in Initial State
Device ID and Channel IDs are initialized, h/w arbitration is enabled (if supported) and specified power-on default values are loaded.
Select Package or any other command (except Clear Initial State or Deselect Package*) that is directed to the Package or a Channel within the Package
Channels within the controller can independently enter and exit their respective Channel Ready and Initial State states. The package selected / deselected state transition always applies to all channels.
Package Ready, Selected, Channel in Initial State
Device ID and Channel IDs are initialized, hardware arbitration (if supported) is enabled or re-enabled and specified power-on default values are loaded or restored.
Any command except Clear Initial State or Deselect Package
Reset Channel command or Asynchronous Entry into Initial State
Clear Initial State command
Deselect Package command
Clear Initial State command
Package Ready, Deselected, Channel in Initial State
Clear Initial State command
Package Ready, Selected, Channel Ready
Deselect Package command*
Typical Operational State
Channel Not Ready
Package Not Ready
These transient states may be entered at any time, from any state or transition between other states. These states will be exited either by returning to the state they were entered from, or by completing the transition between the other states.
Select Package or any other command that is directed to the Package or a Channel within the Package
Deselect Package command
Asynchronous Entry into Initial State
Reset Channel command
Package Ready, Deselected, Channel Ready
Deselect Package command
* A deselected package that receives a Deselect Package command will become temporarily selected until it delivers the response to the command, after which the package returns to being deselected.

1211
1212

1213 **Figure 6 – NC-SI package/channel operational state diagram**

### 6.1.7 State diagram for NC-SI operation with hardware arbitration

Figure 7 shows NC-SI operation in the hardware arbitration mode of operation. This is a sub-set of the general NC-SI operational state diagram (Figure 6) and has been included to illustrate the simplified sequence of package selection when this optional capability is used.


Interface Power Up
Controller Startup / Initialization
Device ID and Channel IDs are initialized, h/w arbitration is enabled and specified power-on default values are loaded or restored.
Power Off
Interface Powered Up, Package & Channel(s) Not Ready *Bypass Mode*
Package Ready, Not Selected, Channel(s) in Initial State
*Any* command other than *Clear Initial State*
Package Ready, Selected, Channel in Initial State
*Reset Channel* command or Asynchronous Entry into Initial State
Channel Not Ready
***Clear Initial State* command**
*Clear Initial State* command
Package Ready, Selected, Channel Ready
This transient state may be entered at any time, from any state or transition between other states. This state will be exited either by returning to the state it was entered from, or by completing the transition between other states.
**Typical operational state**

Channels within the controller (package) can independently enter and exit their respective Channel States.

**Figure 7 – NC-SI operational state diagram for hardware arbitration operation**

While Select and Deselect package commands are not shown in Figure 7, these commands can be used with HW arbitration and will behave as specified in this specification.

Select and Deselect package commands can work together with HW arbitration. If HW arbitration is enabled, a package needs both the HW arbitration token and to be selected in order to transmit on the NC-SI RBT. If either the package is deselected, or the package does not have HW arbitration token, then the package is not allowed to transmit on the NC-SI RBT.

### 6.1.8 Resets

#### 6.1.8.1 Asynchronous entry into Initial State

An Asynchronous Reset event is defined as an event that results in a Channel asynchronously entering the Initial State. This event could occur as a consequence of powering up, a System Reset, a Driver Reset, an internal firmware error, loss of configuration errors, internal hardware errors, and so on. Additionally, it is recommended that any event in the NC that causes a total or partial loss of configuration should be interpreted as an Asynchronous Reset event

Unless otherwise specified, NC-SI configuration settings beyond those required by the Initial State may or may not be preserved following asynchronous entry into the Initial State, depending on the Network Controller implementation.

There is no explicit definition of a Reset for an entire package. However, it is possible that an Asynchronous Reset condition may cause an asynchronous entry into the Initial State for all Channels in a package simultaneously.

#### 6.1.8.2 Synchronous Reset

A Synchronous Reset event on the NC-SI is defined as a Reset Channel command issued by a Management Controller to a Channel. Upon the receipt of this command, the Network Controller shall place the Channel into the Initial State.

Unless otherwise specified, NC-SI configuration settings beyond those required by the Initial State may or may not be preserved following a Synchronous Reset, depending on the Network Controller implementation.

#### 6.1.8.3 Other Resets

Resets that do not affect NC-SI operation are outside the scope of this specification.

### 6.1.9 Network Controller Channel ID

Each channel in the Network Controller shall be physically assigned a Network Controller Channel ID that will be used by the Management Controller to specify which Network Controller channel, of possibly many, it is trying to communicate. The Network Controller Channel ID shall be physically assignable (configured) at system-integration time based on the following specification.

It is the system integrator’s or system designer’s responsibility to correctly assign and provide these identifier values in single- and multi-port Network Controller configurations, and to ensure that Channel IDs do not conflict between devices sharing a common NC-SI RBT interconnect.

The Channel ID field is composed of two subfields, Package ID and Internal Channel ID, as described in Table 2.

**Table 2 – Channel ID format**

| Bits | Field Name | Description |
|---|---|---|
| [7..5] | Package ID | The Package ID is required to be common across all channels within a single Network Controller that share a common NC-SI physical interconnect.<br>The system integrator will typically configure the Package IDs starting from 0 and increasing sequentially for each physical Network Controller.<br>The Network Controller shall allow the least significant two bits of this field to be configurable by the system integrator, with the most significant bit of this field = `0b`. An implementation is allowed to have all 3 bits configurable. |
| [4..0] | Internal Channel ID | The Network Controller shall support Internal Channel IDs that are numbered starting from 0 and increasing sequentially for each channel supported by the Network Controller that is accessible by the Management Controller through the NC-SI using NC-SI commands.<br>An implementation is allowed to support additional configuration options for the Internal Channel ID as long as the required numbering can be configured.<br>An Internal Channel ID value of `0x1F` applies to the entire Package. |

Channel IDs shall be completely decoded. Aliasing between values is not allowed (that is, the Network Controller is not allowed to have multiple IDs select the same channel on a given Sideband Interface).

Once configured, the settings of the Package ID and Internal Channel ID values shall be retained in a non-volatile manner. That is, they shall be retained across power-downs of the Sideband Interface and shall not be required to be restored by the Management Controller for NC-SI operation. This specification does not define the mechanism for configuring or retaining the Package ID or the Internal Channel ID (if configurable). Some implementations may use pins on the Network Controller for configuring the IDs, other implementations may use non-volatile storage logic such as electrically erasable memory or FLASH, while others may use a combination of pins and non-volatile storage logic.

## 6.1.10 Configuration-related settings

### 6.1.10.1 Package-specific operation

There are some NC-SI configuration settings that are package-specific:

- the enable/disable settings for hardware arbitration
- NC-SI flow control
- Package-related AENs

There may also be NC configuration settings that are controlled by NC-SI Commands addressed to the package. These commands specify this requirement in their command description.

Hardware arbitration is enabled or disabled through a parameter that is delivered using the Select Package command. If hardware arbitration is enabled on all Network Controller packages on the NC-SI RBT, more than one package can be in the Selected state simultaneously. Otherwise, only one package is allowed to be in the Selected state at a time to prevent electrical buffer conflicts (buffer fights) that can occur from more than one package being allowed to drive the bus.

NC-SI flow control is enabled or disabled using the Set NC-SI Flow Control command. The flow control setting applies to all channels in the package.

Package-specific commands should only be allowed and executed when the Internal Channel ID field is set to `0x1F`.

There are some package-level AENs to allow the NC to alert the MC of controller-level events.

#### 6.1.10.2 Channel-specific operation

Channel-specific commands should only be allowed to be executed when the Internal Channel ID field is set to a value other than `0x1F`. Channel-specific commands with Invalid Channel IDs are not allowed (see clause 6.8.2.1).

Table 3 shows the major categories of configuration settings that control channel operation when a channel is in the Channel Ready state. Channels that are not operating in Pass-through mode may not support Pass-through-related settings.

**Table 3 – Channel Ready state configuration settings**

| Setting/Configuration Category | Description |
|---|---|
| “Channel Enable” settings | The Enable Channel and Disable Channel commands are used to control whether the channel is allowed to asynchronously transmit unrequested packets (AEN and Pass-through packets) through the NC-SI interface whenever the package is Selected. Note that channels are always allowed to transmit responses to commands sent to the channel. |
| “Channel Configuration” settings | Version 1.2 adds a number of commands for configuration setting of channels and their partitions (if supported)  See Table 19 |
| Pass-through Transmit Enable settings | The Enable Channel Network TX command is used to enable the channel to transmit any Pass-through packets that it receives through the NC-SI onto the network, provided that the source MAC address in those packets matches the Network Controller settings. Correspondingly, the Disable Channel Network TX command is used to direct the controller not to transmit Pass-through packets that it receives onto the network. |
| AEN Enable settings | The AEN Enable command is used to enable and disable the generation of the different AENs supported by the Network Controller. |
| MAC Address Filter settings and control | The Set MAC Address, Enable Broadcast Filter, and Enable Global Multicast Filter commands are used to configure the filters for unicast, broadcast, and multicast addresses that the controller uses in conjunction with the VLAN Filter settings for filtering incoming Pass-through packets. |
| VLAN Filter settings and control | The Set VLAN Filter command is used to configure VLAN Filters that the controller uses in conjunction with the MAC Address Filters for filtering incoming Pass-through packets. The Enable VLAN and Disable VLAN commands are used to configure VLAN filtering modes and enable or disable whether VLAN filtering is used. |

### 6.1.11 Transmitting Pass-through packets from the Management Controller

Packets not recognized as command packets (that is, packets without the NC-SI Ethertype) that are received on the Network Controller’s NC-SI interface shall be assumed to be Pass-through packets provided that the source MAC Address matches one of the unicast MAC addresses settings (as configured by the Set MAC Address command) for the channel in the Network Controller, and will be forwarded for transmission to the corresponding external network interface if Channel Network TX is enabled.

### 6.1.12 Receiving Pass-through packets for the Management Controller

The Management Controller has control over and responsibility for configuring packet-filtering options, such as whether broadcast, multicast, or VLAN-tagged packets are accepted. Depending on the filter configurations, after the channel has been enabled, any packet that the Network Controller receives for the Management Controller shall be forwarded to the Management Controller through the NC-SI interface.

### 6.1.13 Pass-through operation in multiple medium implementations

Pass-through operation is not restricted to certain physical interfaces, but a NC-SI channel shall support Pass-through on at most one physical interface at a time.

### 6.1.14 Startup sequence examples

#### 6.1.14.1 Overview

The following clauses show possible startup sequences that may be used by the Management Controller to start NC-SI operation. Depending upon the specific configuration of each system, there are many possible variations of startup sequences that may be used, and these examples are intended for reference only.

#### 6.1.14.2 Typical non-hardware arbitration specific startup sequence

The following sequence is provided as an example of one way a Management Controller can start up NC-SI operation. This sequence assumes that the Management Controller has no prior knowledge of how many Network Controllers are present on RBT, or what capabilities those controllers support. Note that this is not the only possible startup sequence. Alternative sequences can also be used to start up NC-SI operation. Some steps may be skipped if the Management Controller has prior knowledge of the Network Controller capabilities, such as whether Network Controllers are already connected and enabled for hardware arbitration.

1) Power up

   The NC-SI is powered up (refer to clause 10.2.8 for the specification of this condition). The Network Controller packages are provided a Network Controller Power Up Ready Interval during which they can perform internal firmware startup and initialization to prepare their NC-SI to accept commands. The Management Controller first waits for the maximum Network Controller Power Up Ready Interval to expire (refer to Table 279). At this point, all the Network Controller packages and channels should be ready to accept commands through the NC-SI. (The Management Controller may also start sending commands before the Network Controller Power Up Ready Interval expires but will have to handle the case that Network Controller devices may be in a state in which they are unable to accept or respond to commands.)

2) Discover package

   The Management Controller issues a Select Package command starting with the lowest Package ID (see clause 8.4.5 for more information). Because the Management Controller is assumed to have no prior knowledge of whether the Network Controller is enabled for hardware arbitration, the Select Package command is issued with the Hardware Arbitration parameter set to ‘disable’.

   If the Management Controller receives a response within the specified response time, it can record that it detected a package at that ID. If the Management Controller does not receive a response, it is recommended that the Management Controller retry sending the command. Three total tries are typical. (This same retry process should be used when sending all commands to the Network Controller and will be left out of the descriptions in the following steps.) If the retries fail, the Management Controller can assume that no Network Controller is at that Package ID and can immediately repeat this step 2) for the next Package ID in the sequence.

3) Discover and get capabilities for each channel in the package

The Management Controller can now discover how many channels are supported in the Network Controller package and their capabilities. To do this, the Management Controller issues the Clear Initial State command starting from the lowest Internal Channel ID (which selects a given channel within a package). If it receives a response, the Management Controller can then use the Get Version ID command to determine NC-SI specification compatibility, and the Get Capabilities command to collect information about the capabilities of the channel. The Management Controller can then repeat this step until the full number of internal channels has been discovered. (The Get Capabilities command includes a value that indicates the number of channels supported within the given package.)

NOTE The *NC-SI Specification* requires Network Controllers to be configurable to have their Internal Channel IDs be sequential starting from 0. If it is known that the Network Controller is configured this way, the Management Controller needs only to iterate sequentially starting from Internal Channel ID = 0 up to the number of channels reported in the first Get Capabilities response.

The Management Controller should temporarily retain the information from the Get Capabilities command, including the information that reports whether the overall package supports hardware arbitration. This information is used in later steps.

4) Repeat steps 2 and 3 for remaining packages

The Management Controller repeats steps 2) and 3) until it has gone through all the Package IDs.

IMPORTANT: Because hardware arbitration has not been enabled yet, the Management Controller shall issue a Deselect Package command to the present Package ID before issuing the Select Package command to the next Package ID. If hardware arbitration is not being used, only one package can be in the Selected state at a time. Otherwise, hardware electrical buffer conflicts (buffer fights) will occur between packages.

5) Initialize each channel in the package

Based on the number of packages and channels that were discovered, their capabilities, and the desired use of Pass-through communication, the Management Controller can initialize the settings for each channel. This process includes the following general steps for each package:

a) Issue the Select Package command.

b) For each channel in the package, depending on controller capabilities, perform the following actions. Refer to individual command descriptions for more information.

- Use the Set MAC Address command to configure which unicast and multicast addresses are used for routing Pass-through packets to and from the Management Controller.
- Use the Enable Broadcast Filter command to configure whether incoming broadcast Pass-through packets are accepted or rejected.
- Use the Enable Global Multicast Filter command to configure how incoming multicast Pass-through packets are handled based on settings from the Set MAC Address command.
- Use the Set VLAN Filter and Enable VLAN Filters commands to configure how incoming Pass-through packets with VLAN Tags are handled.
- Use the Set NC-SI Flow Control command (if supported) to configure how Ethernet Pause Frames are used for flow control on RBT. Set NC-SI Flow Control is a package command and only needs to be issued once.
- Use the AEN Enable command to configure what types of AEN packets the channel should send out on the NC-SI.

- Use the Enable Channel Network TX command to configure whether the channel is enabled to deliver Pass-through packets from the NC-SI to the network (based on the MAC address settings) or is disabled from delivering any Pass-through packets to the network.

c) Issue the Deselect Package command.

6) Start Pass-through packet and AEN operation on the channels

The channels should now have been initialized with the appropriate parameters for Pass-through packet reception and AEN operation. Pass-through operation can be started by issuing the Enable Channel command to each channel that is to be enabled for delivering Pass-through packets or generating AENs through the NC-SI interface.

NOTE: If hardware arbitration is not operational and it is necessary to switch operation over to another package, a Deselect Package command shall be issued to the presently selected package before a different package can be selected. Deselecting a package blocks all output from the package. Therefore, it is not necessary to issue Disable Channel commands before selecting another package. There is no restriction on enabling multiple channels within a package.

#### 6.1.14.3 Hardware arbitration-specific startup sequence

This clause applies when multiple NCs are used by the MC. This clause only applies to the NC-SI over RBT binding.

The following is an example of the steps that a Management Controller may perform to start up NC-SI operation when Hardware Arbitration is specifically known to be used, present, and enabled on all Network Controllers. This example startup sequence assumes a high level of integration where the Management Controller knows the Network Controllers support and default to the use of Hardware Arbitration on startup but does not have prior knowledge of how many Network Controllers are present on RBT, or the full set of capabilities those controllers support, so discovery is still required.

Although other startup examples may show a specific ordering of steps for the process of discovering, configuring and enabling channels, the Management Controller has almost total flexibility in choosing how these steps are performed once a channel in a package is discovered. In the end, it would be just as valid for a Management Controller to follow a breadth-first approach to discovery steps as it would be to follow a depth-first approach where each channel that is discovered is fully initialized and enabled before moving to the next.

**1) Power up**

No change from other startup scenarios.

**2) Discovery**

The process of discovery consists of identifying the number of packages that are available, the number of channels that are available in each package, and for each channel, the capabilities that are provided for Management Controller use. Because, in this startup scenario, the Management Controller knows Hardware Arbitration is used, it is not required to use the ***Select Package*** and ***Deselect Package*** commands for discovery but may elect to just use the ***Clear Initial State*** command for this purpose instead.

In this startup scenario, Packages and Channels are discovered by sending the ***Clear Initial State*** command starting with the lowest Package ID and Internal Channel ID, then waiting for, and recording, the response event as previously described. Internal channel IDs are required to be numbered sequentially starting with 0, so when the Management Controller does not receive a response to repeated attempts at discovery, it knows this means no additional channels exist in the current package. If this happens when the internal channel ID is 0, the Management Controller knows a package is not available at the current package ID, and it continues with the next package ID in sequence. If the Management Controller receives a response to the ***Clear Initial State*** command, it records that the channel and package are available, and continues discovery.

During discovery, the Management Controller should interrogate the capabilities of each channel found to be available in each package by sending the ***Get Capabilities*** command appropriate package and Internal channel ID values. However, it does not matter whether this is done as the very next step in the discovery process or performed for each channel after all packages and channels have been discovered, just as long as the Management Controller does interrogate each channel.

**3) Configure each channel and enable pass-through**

Once the existence of all packages and channels, and the capabilities of each channel, have been discovered and recorded, the Management Controller shall initialize and enable each channel as needed for use. The details of these steps remain essentially the same as have been previously stated, except to note that there are no restrictions on how they are performed. What this means is that the MC may perform these steps in any order across the channels in each package as it sees fit. The MC may fully initialize and enable each channel in each package one at a time or perform the same step on each channel in sequence before moving on to the next, or in a different order. The specific order of steps is not dictated by this specification.

#### 6.1.14.4 Summary of scheme for the MC without prior knowledge of hardware arbitration

The following scheme describes the case when the MC does not have a priori knowledge of the hardware arbitration support across multiple NCs.

1. For each available NC,

    a. The MC checks whether a device supports the HW arbitration, using “**Get Capabilities**” command (this implicitly selects the package).

    b. The MC issues “Deselect Package” for the NC (needed as at this stage we do not know whether all the devices support HW arbitration).

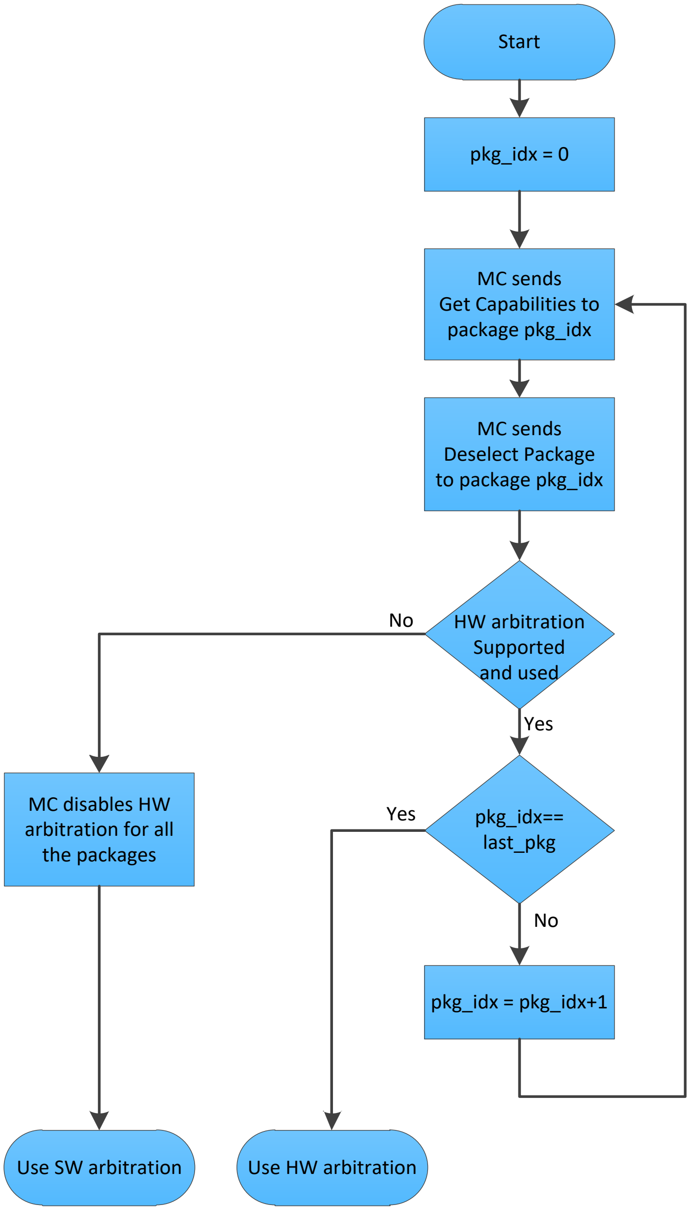
2. If (all NCs support HW arbitration and HW arbitration is used by all NCs), then

    the MC assumes that HW arbitration is active because according to clause 6.2.4 “set hardware arbitration (if supported) to *enabled* on Interface Power Up only”, and the MC can “Select” any number of packages at the same time.

   Otherwise (at least one NC reports that HW arbitration is not supported, or at least one NC reports that HW arbitration is not used, or at least one NC cannot report its support level) then

    HW arbitration is **not** active, and the MC can “Select” only single package at the any time.

    The MC configures every NC to disable HW arbitration, using the “***Select Package***” command.


Start
pkg_idx = 0
MC sends Get Capabilities to package pkg_idx
MC sends Deselect Package to package pkg_idx
HW arbitration Supported and used
No
Yes
MC disables HW arbitration for all the packages
pkg_idx== last_pkg
Yes
No
pkg_idx = pkg_idx+1
Use SW arbitration
Use HW arbitration

1477

1478 **Figure 8 – MC steps when the MC does not have prior knowledge of hardware arbitration**

## 6.2 NC-SI traffic types

### 6.2.1 Overview

Two types of traffic are defined by NC-SI, based on the network fabric type: Pass-through traffic and Control traffic.

- Pass-through traffic consists of packets that are transferred between the external network interface and the Management Controller using the Sideband Interface.
- Control traffic consists of commands (requests) and responses that support the inventory, configuration and control of the Network Controller, the Sideband Interface and Pass-through operation of the Network Controller, and AENs that support reporting various events to the Management Controller.

### 6.2.2 Command protocol

#### 6.2.2.1 Overview

Commands are provided to allow a Management Controller to initialize, control, and regulate Management Controller packet flow across the sideband interface, configure channel filtering, and to interrogate the operational status of the Network Controller. As interface master, the Management Controller is the initiator of all commands, and the Network Controller responds to commands, but may also generated AENs if enabled.

#### 6.2.2.2 Instance IDs

The command protocol uses a packet field called the Instance ID (IID). IID numbers are 8-bit values that shall range from `0x01` to `0xFF`. IIDs are used to uniquely identify instances of a command, to improve the robustness of matching responses to commands, and to differentiate between new and retried commands. The Network Controller that receives a command handles the IID in the following ways:

- It returns the IID value from the command in the corresponding response.
- If the IID is the same as the IID for the previous command, it recognizes the command as a ‘retried’ command rather than as a new instance of the command. It is expected that the ‘retried’ command contains the same command type value in the Control Packet Type field. The NC behavior when a ‘retried’ command type does not match the original command type is outside the scope of this specification.
- If a retried command is received, the Network Controller shall return the previous response. Depending on the command, the Network Controller can accomplish this either by holding the previous response data so that it can be returned, or, if re-executing the command has no side effects (that is, the command is idempotent*)*, by re-executing the command operation and returning that response.
- If the command IID is the same as the IID for the previous command, and the Poll Indication is set, the NC recognizes the command as a ‘polling’ command rather than as a new instance of the command.
    - When polling, the MC is expected to use the command type value of the original command in the Control Packet Type field. If there was no command in progress, the NC shall fail the ‘polling’ command and respond with an error. When the NC fails the ‘polling’ command, the outcome of the original command is indeterminate and is outside the scope of this specification.

  - If a command with Poll Indication set is received and the original command has been completed, then the Network Controller shall return the response of the completed command.
  - If it is still processing the command, it shall return a “Delayed Response” reason code and optionally recommend a next polling time interval.
- When an IID value is received that is different from the one for the previous command, the Network Controller executes the command as a new command.
- When the NC-SI Channel first enters the Initial State, it shall clear any record of any prior requests. That is, it assumes that the first command after entering the Initial State is a new command and not a retried command, regardless of any IID that it may have received before entering the Initial State.

Thus, for single-threaded operation with idempotent commands, a responding Network Controller can simply execute the command and return the IID in the response that it received in the command. If it is necessary to not execute a retried command, the responding controller can use the IID to identify the retried command and return the response that was delivered for the original command.

The Management Controller that generates a command handles the IID in the following ways:

- The IID changes for each new instance of a command.
- If a command needs to be retried, the Management Controller uses the same value for the IID that it used for the initial command.
- The Management Controller can optionally elect to use the IID to provide additional confirmation that the response is being returned for a particular command.

Because an AEN is not a response, an AEN always uses a value of `0x00` for its IID.

NOTE: The Instance ID mechanism can be readily extended in the future to support multiple controllers and multiple outstanding commands. This extension would require having the responder track the IID on a per command and per requesting controller basis. For example, a retried command would be identified if the IID and command matched the IID and command for a prior command for the given originating controller’s ID. That is, a match is made with the command, originating controller, and IID fields rather than on the IID field alone. A requester that generates multiple outstanding commands would correspondingly need to track responses based on both command and IID to match a given response with a given command. IIDs need to be unique for the number of different commands that can be concurrently outstanding.

### 6.2.2.3 Single-threaded operation

The Network Controller is required to support NC-SI commands only in a single-threaded manner. That is, the Network Controller is required to support processing only one command at a time and is not required to accept additional commands until after it has sent the response to the previous one.

Therefore, the Management Controller should issue NC-SI commands in a single-threaded manner. That is, the Management Controller should have only one command outstanding to a given Network Controller package at a time. Upon sending an NC-SI command packet, and before sending a subsequent command, the Management Controller should wait for the corresponding response packet to be received or a command timeout event to occur before attempting to send another command. For the full descriptions of command timeout, see clause 6.8.3.2.

NOTE: While NC implementations are only required to support single-threaded operations, they may choose to support more than one outstanding command. The use of unique IIDs is essential to properly match multiple outstanding commands and responses in such implementations.

#### 6.2.2.4 Responses

The Network Controller shall process and acknowledge each validly formatted command received at the NC-SI interface by formatting and sending a valid response packet to the Management Controller through the NC-SI interface.

To allow the Management Controller to match responses to commands, the Network Controller shall copy the IID number of the Command into the Instance ID field of the corresponding response packet.

To allow for retransmission and error recovery, the Network Controller may re-execute the last command or maintain a copy of the response packet most recently transmitted to the Management Controller through its sideband interface. This “previous” response packet shall be updated every time a new response packet is transmitted to the Management Controller by replacing it with the one just sent.

The Network Controller shall return a “Command Unsupported” response code with an “Unknown Command Type” reason code for any command (standard or OEM) that the Network Controller does not support or recognize. If a command cannot be executed due to the processing of others, the response code Command Unavailable shall be returned.

#### 6.2.2.5 Response and post-response processing

Typically, a Network Controller completes a requested operation before sending the response. In some situations, however, it may be useful for the controller to be allowed to queue up the requested operation and send the response assuming that the operation will complete correctly (for example, when the controller is requested to change link configuration). The following provisions support this process:

- A Network Controller is allowed to send a response before performing the requested action if the command is expected to complete normally and all parameters that are required to be returned with the response are provided.
- Temporal ordering of requested operations shall be preserved. For example, if one command updates a configuration parameter value and a following command reads back that parameter, the operation requested first shall complete so that the following operation returns the updated parameter.
- Under typical operation of the Network Controller, responses should be delivered within the Normal Execution Interval (T5) (see Table 279).
- Unless otherwise specified, all requested operations shall complete within the Asynchronous Reset/Asynchronous Not Ready interval (T6) following the response.
- If the Network Controller channel determines that the requested operation or configuration change has not been completed correctly after sending the response, the channel shall enter the Initial State.
- If the command response is dependent on the execution of the command and the command response cannot be provided within Normal Execution Interval (T5), then a “Delayed Response” response code may be returned. In this case, the MC can poll the command later with the “Poll Indication” set to retrieve the response. The decision on when the MC polls again can be based on one of the following criteria:
    - A fixed delay. In this case a delay greater than T5 is recommended.
    - If provided, based on the “recommended next polling time” in the original response
    - If the AEN is enabled, based on reception of a “Delayed Response Ready AEN”

When using delayed responses, the NC shall complete the command processing within T14 sec.

#### 6.2.2.6 NC-SI traffic ordering

This specification does not require any ordering between AENs, NC-SI responses, and NC-SI Pass-through packets. Specific transport binding specifications may require ordering between AENs, NC-SI responses, and NC-SI Pass-through packets.

## 6.3 Link configuration and control

### 6.3.1 Link Configuration

The Network Controller provides commands to allow the Management Controller to specify the auto-negotiation, link speed, duplex settings, FEC algorithm, link training, SerDes lane configuration, and so on to be used on the network interface. For more information, see clause 8.4.21.

The Management Controller should make link configuration changes only when the host network driver is absent or non-operational.

### 6.3.2 Link Status

The Network Controller provides a Get Link Status command to allow the Management Controller to interrogate the configuration and operational status of the primary links. The Management Controller may issue the Get Link Status command regardless of OS operational status.

## 6.4 Frame filtering for Pass-through mode

### 6.4.1 Overview

The Network Controller provides the option of configuring various types of filtering mechanisms for the purpose of controlling the delivery of received Ethernet frames to the Management Controller. These options include VLAN Tag filter, L2 address filters, MAC address support, and limited frame filtering using L3, L4 protocol header fields. All frames that pass frame filtering are forwarded to the Management Controller over the Sideband Interface. Refer to RFC2373, RFC2461, and RFC3315 for IPv6-related definitions.

### 6.4.2 Multicast filtering

The Network Controller may provide commands to allow the Management Controller to enable and disable global filtering of all multicast packets. The Network Controller may optionally provide one or more individual multicast filters, as well as DHCP v6, IPv6 Neighbor Advertisement, IPv6 Router Advertisement, IPv6 Neighbor Solicitation, IPv6 MLD, mDNSv4, mDNSv6 and LLDP filters.

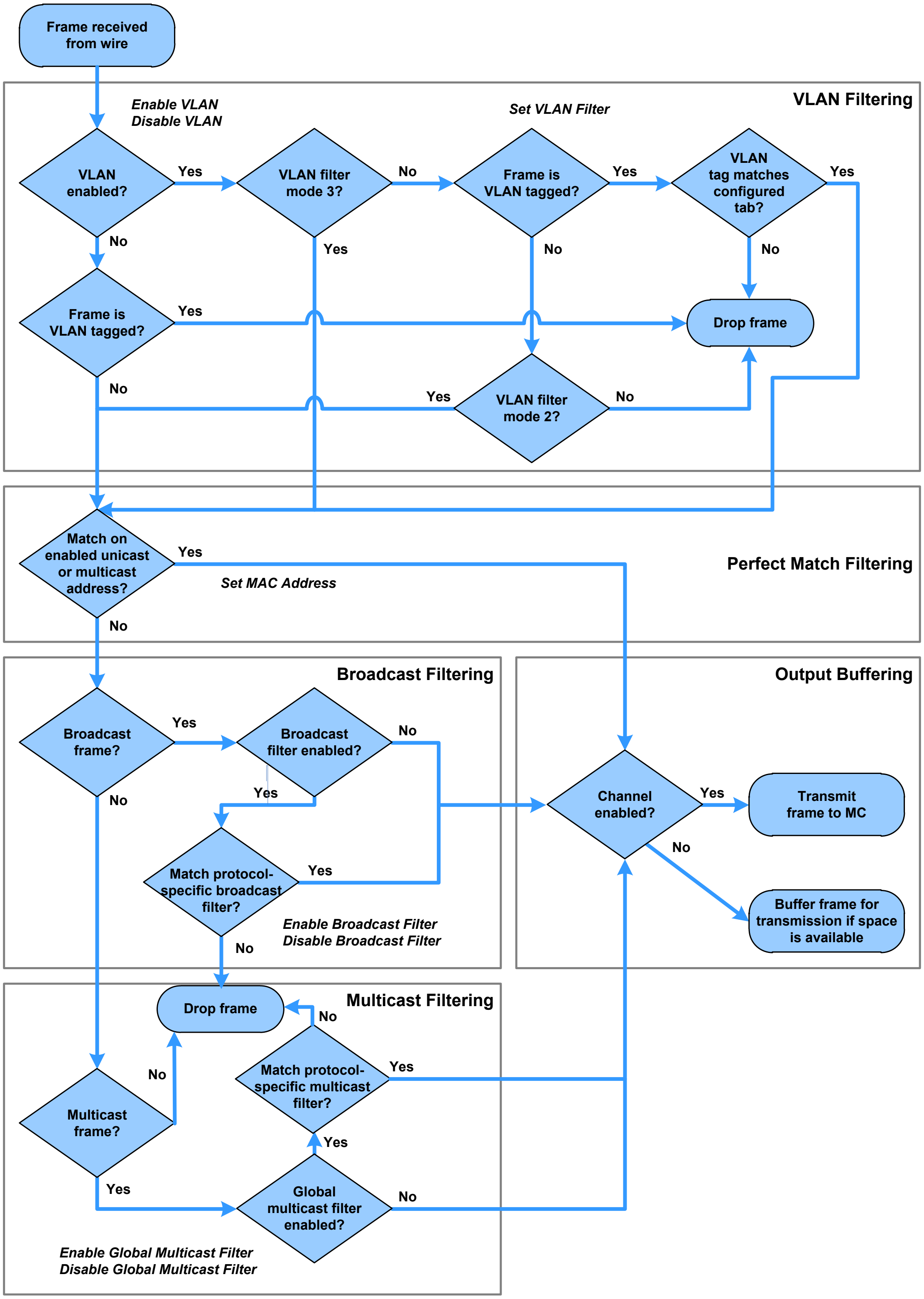
### 6.4.3 Broadcast filtering

The Network Controller provides commands to allow the Management Controller to enable and disable forwarding of Broadcast and ARP packets. The Network Controller may optionally support selective forwarding of broadcast packets for specific protocols, such as DHCP (see RFC2131) and NetBIOS.

### 6.4.4 VLAN filtering

The Network Controller provides commands to allow the Management Controller to enable and disable VLAN filtering, configure one or more VLAN Filters, and to configure VLAN filtering modes.

Figure 9 illustrates the flow of frame filtering. Italicized text in the figure is used to identify NC-SI command names.



1642

1643 **Figure 9 – NC-SI packet filtering flowchart**

## 6.5 Output buffering behavior

There are times when the NC is not allowed to transmit Pass-through, AEN, or Control Packets onto the Sideband Interface.

The NC should buffer Pass-through frames to be transmitted to the MC under any of the following conditions:

- The package is deselected.
- For a channel within a package while that channel is disabled.
- When the hardware arbitration is enabled, and the NC does not have the token to transmit frames to the MC.

The NC may buffer AENs to the MC under any of the above conditions.

Control Packets (responses) are buffered when hardware arbitration is enabled, and the NC does not have the token to transmit frames to the MC.

Additionally, while an NC-SI channel is in the initial state, previously received Pass-through frames and AENs may or may not be buffered. This behavior is outside the scope of this specification.

## 6.6 NC-SI flow control

The Network Controller may provide commands to enable flow control on the RBT interface between the Network Controller and the Management Controller. The NC-SI flow control behavior follows the PAUSE frame behavior as defined in the IEEE 802.3. Flow control is configured using the Set NC-SI Flow command (see clause 8.4.41).

When enabled for flow control, a channel may direct the package to generate and renew 802.3x (XOFF) PAUSE Frames for a maximum interval of T12 for a single congestion condition. If the congestion condition remains in place after a second T12 interval expires, the congested channel shall enter the Initial State and remove its XOFF request to the package. Note that some implementations may have shared buffering arrangements where all channels within the package become congested simultaneously. Also note that if channels become congested independently, the package may not immediately go into the XON state after T12 if other channels within the package are still requesting XOFF.

## 6.7 Asynchronous Event Notification

### 6.7.1 Overview

Asynchronous Event Notification (AEN) packets enable the Network Controller to deliver unsolicited notifications to the Management Controller when certain status changes that could impact interface operation occur in the Network Controller. Because the NC-SI is a small part of the larger Network Controller, its operation can be affected by a variety of events that occur in the Network Controller. These events include link status changes, OS driver loads and unloads, and chip resets. This feature defines a set of notification packets that operate outside of the established command-response mechanism.

Control over the generation of the AEN packets is achieved by control bits in the AEN Enable command. Each type of notification is optional and can be independently enabled by the Management Controller.

AENs are not acknowledged, and there is no protection against the possible loss of an AEN packet. Each defined event has its own AEN packet. Because the AEN packets are generated asynchronously by the Network Controller, they cannot implement some of the features of the other Control Packets. AEN packets leverage the general packet format of Control Packets.

- The originating Network Controller shall fill in the Channel ID (Ch. ID) field as defined in clause 6.1.9 in the AEN header to identify the source of notification.

- The IID field in an AEN shall be set to `0x00` to differentiate it from a response or command packet.
- The Network Controller shall copy the AEN MC ID field from the AEN Enable command into the MC ID field in every AEN sent to the Management Controller.

### 6.7.2 AEN handling in multiple medium implementations

Implementations that use NC-SI over physical interfaces other than RBT and enable Asynchronous Event Notifications (AEN) on those other media used for MCTP shall comply with the requirements in DSP0261.

AENs that are enabled via RBT are specific to RBT-active operation and any AEN that is subsequently generated is only delivered over RBT and then only when RBT is active (maintained or restored operation).

AEN generation is suppressed and not cached when the media on which it was enabled is not active.

## 6.8 Error handling

### 6.8.1 Overview

This clause describes the error-handling methods that are supported over the NC-SI. Two types of error-handling methods are defined:

- Synchronous Error Handling
- Errors that trigger Asynchronous Entry into the Initial State

Synchronous Error Handling occurs when an Error (non-zero) Response/Reason Code is received in response to a command issued by the Management Controller. For information about response and reason codes, see clause 8.2.4.1.

Asynchronous Entry into the Initial State Error Handling occurs when the Network Controller asynchronously enters the Initial State because of an error condition that affects NC-SI configuration or a failure of a command that was already responded to. For more information, see clause 6.1.8.1.

### 6.8.2 Transport errors

#### 6.8.2.1 Dropped Control Packets

A Network Controller with an active interface shall drop Control Packets received on the NC-SI interface under the following conditions:

- The packet has an invalid Frame Check Sequence (FCS) value.
- Frame length does not meet IEEE 802.3 requirements (except for OEM commands, where accepting larger packets may be allowed as a vendor-specific option).
- The packet checksum (if provided) is invalid.
- The NC-SI Channel ID value in the packet does not match the expected value.
- The Network Controller does not have resources available to accept the packet.
- The Network Controller receives a command packet with an incorrect header revision.
- Control Packets may also be dropped if an event that triggers Asynchronous Entry into the Initial State causes packets to be dropped during the transition..

#### 6.8.2.2 Pass-through packet errors

Handling of Pass-through packet errors, other than logging statistics, is out of scope of this specification.

### 6.8.3 Missing responses

#### 6.8.3.1 Overview

There are typical scenarios in which the Management Controller does not receive the response to a command:

- The Network Controller dropped the command and thus never sent the response.
- The response was dropped by the Management Controller (for example, because of a CRC error in the response packet).
- The Network Controller is in the process of being reset or is disabled.

The Management Controller can detect a missing response packet as the occurrence of an NC-SI command timeout event.

#### 6.8.3.2 Command timeout

The Management Controller may detect missing responses by implementing a command timeout interval. The timeout value chosen by the Management Controller shall not be less than Normal Execution Interval, T5. Upon detecting a timeout condition, the Management Controller should not make assumptions on the state of the unacknowledged command (for example, the command was dropped, or the response was dropped), but should retransmit (retry) the previous command using the same IID it used in the initial command.

The Management Controller should try a command at least three times before assuming an error condition in the Network Controller.

It is possible that a Network Controller could send a response to the original command at the same time a retried command is being delivered. Under this condition, the Management Controller could get more than one response to the same command. Thus, the Management Controller should be capable of determining that it has received a second instance of a previous response packet. Dropped commands may be detected by the Management Controller as a timeout event waiting for the response.

#### 6.8.3.3 Handling dropped commands or missing responses

To recover from dropped commands or missing responses, the Management Controller can retransmit the unacknowledged command packet using the same IID that it used for the initial command.

The Network Controller shall be capable of reprocessing retransmitted (retried) commands without error or undesirable side effects. The Network Controller can determine that the command has been retransmitted by verifying that the IID is unchanged from the previous command.

### 6.8.4 Detecting Pass-through traffic interruption

The Network Controller might asynchronously enter the Initial State because of a reset or other event. In this case, the Network Controller stops transmitting Pass-through traffic on the RXD lines. Similarly, Pass-through traffic sent to the Network Controller may be dropped. If the Management Controller is not in the state of sending or receiving Pass-through traffic, it may not notice this condition. Thus, the Management Controller should periodically issue a command to the Network Controller to test whether the Network Controller has entered the Initial State. How often this testing should be done is a choice of the Management Controller.

## 6.9 Support for additional network fabrics

### 6.9.1 FC support

NCs that support Fibre Channel connectivity can be inventoried, configured, and monitored. Fibre Channel-specific link speed, link status, boot configuration and statistics commands are provided. Fibre Channel over Ethernet (FCoE) support is also defined for Ethernet NCs that support it.

### 6.9.2 InfiniBand Support

NCs that support InfiniBand connectivity can be inventoried, configured, and monitored. InfiniBand-specific link speed, link status and statistics commands and Pass-through mode support are provided.

## 6.10 PLDM and SPDM transport

NC-SI over RBT can be used to transport SPDM or PLDM messages. This transport supports the following modes:

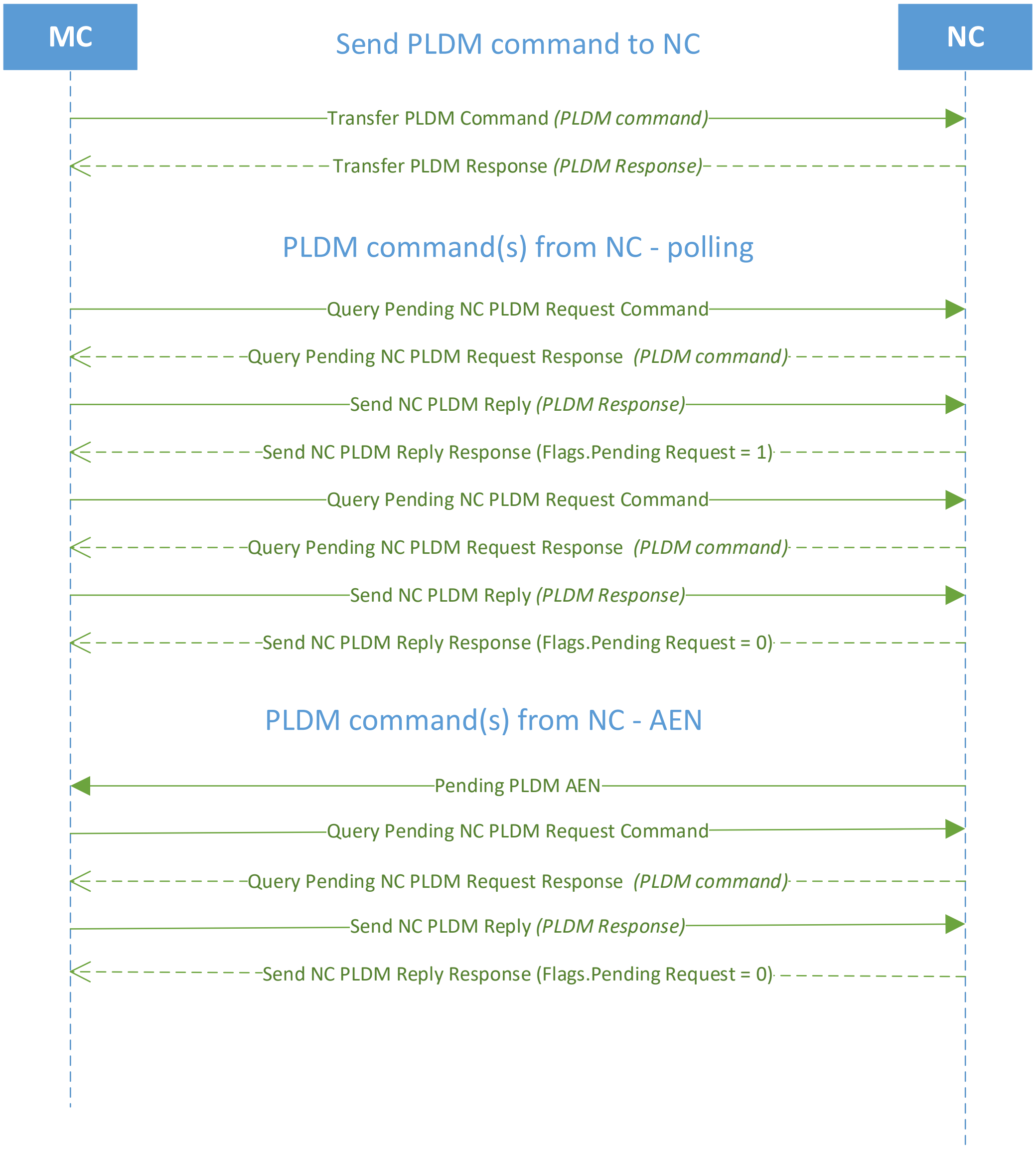
- MC sends PLDM and/or SPDM commands to the NC.
- MC polls the NC for PLDM and/or SPDM commands originating at the NC.
- The NC indicates through an AEN that a PLDM/SPDM command is available for retrieval.

The following commands are used to implement an RBT binding for these messages:

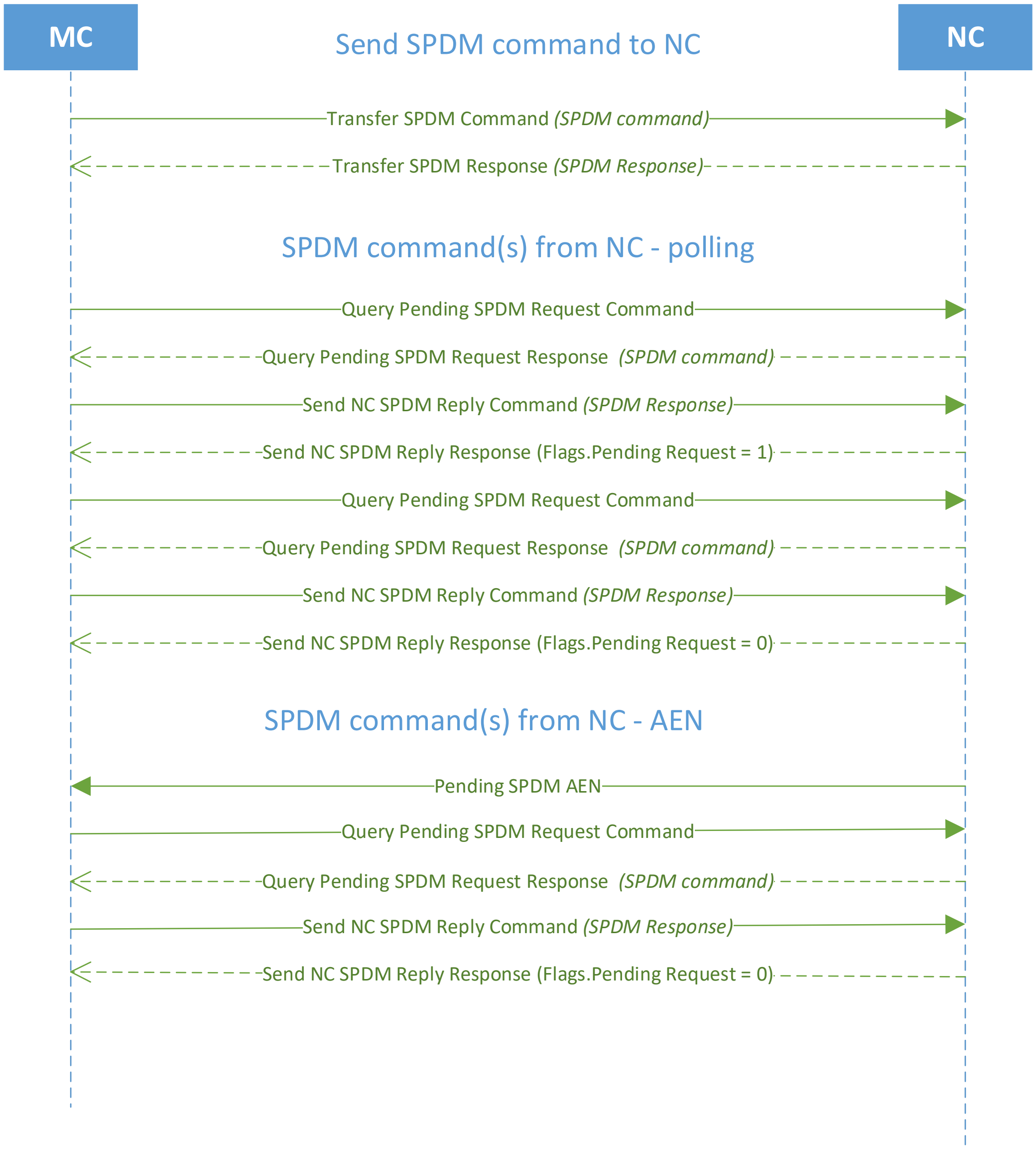
**Table 4 – Commands for RBT binding**

| Command Description | PLDM Commands over RBT | SPDM Commands over RBT |
|---|---|---|
| Send command from MC | PLDM | SPDM |
| Poll for NC command | Query Pending NC PLDM Request | Query Pending SPDM Request |
| Respond to NC command | Send NC PLDM Reply | Send NC SPDM Reply |

The PLDM and SPDM command flows are described in the UML diagrams below.


MC
Send PLDM command to NC
NC
Transfer PLDM Command (PLDM command)
Transfer PLDM Response (PLDM Response)
PLDM command(s) from NC - polling
Query Pending NC PLDM Request Command
Query Pending NC PLDM Request Response (PLDM command)
Send NC PLDM Reply (PLDM Response)
Send NC PLDM Reply Response (Flags.Pending Request = 1)
Query Pending NC PLDM Request Command
Query Pending NC PLDM Request Response (PLDM command)
Send NC PLDM Reply (PLDM Response)
Send NC PLDM Reply Response (Flags.Pending Request = 0)
PLDM command(s) from NC - AEN
Pending PLDM AEN
Query Pending NC PLDM Request Command
Query Pending NC PLDM Request Response (PLDM command)
Send NC PLDM Reply (PLDM Response)
Send NC PLDM Reply Response (Flags.Pending Request = 0)

1780


MC
Send SPDM command to NC
NC
Transfer SPDM Command (SPDM command)
Transfer SPDM Response (SPDM Response)
SPDM command(s) from NC - polling
Query Pending SPDM Request Command
Query Pending SPDM Request Response (SPDM command)
Send NC SPDM Reply Command (SPDM Response)
Send NC SPDM Reply Response (Flags.Pending Request = 1)
Query Pending SPDM Request Command
Query Pending SPDM Request Response (SPDM command)
Send NC SPDM Reply Command (SPDM Response)
Send NC SPDM Reply Response (Flags.Pending Request = 0)
SPDM command(s) from NC - AEN
Pending SPDM AEN
Query Pending SPDM Request Command
Query Pending SPDM Request Response (SPDM command)
Send NC SPDM Reply Command (SPDM Response)
Send NC SPDM Reply Response (Flags.Pending Request = 0)

1781
1782

# 7 Arbitration in configurations with multiple Network Controller packages

## 7.1 Overview

This clause applies to NC-SI over RBT only.

More than one Network Controller package on a RBT interface can be enabled for transmitting packets to the Management Controller. This specification defines two mechanisms to accomplish Network Controller package arbitration operations. One mechanism uses software commands provided by the Network Controller for the Management Controller to control whose turn it is to transmit traffic. The other mechanism uses hardware arbitration to share the single RBT bus. Implementations are required to support command-based Device Selection operation; the hardware arbitration method is typically desired but is optional.

## 7.2 Multi-controller RBT

Figure 10 is a simplified block diagram of the Sideband Interface being used in a multi-drop configuration. The RMII (upon which NC-SI RBT is based) was originally designed for use as a point-to-point interconnect. Accordingly, only one party can transmit data onto the bus at any given time. There is no arbitration protocol intrinsic in the RMII specification to support managing multiple transmitters.

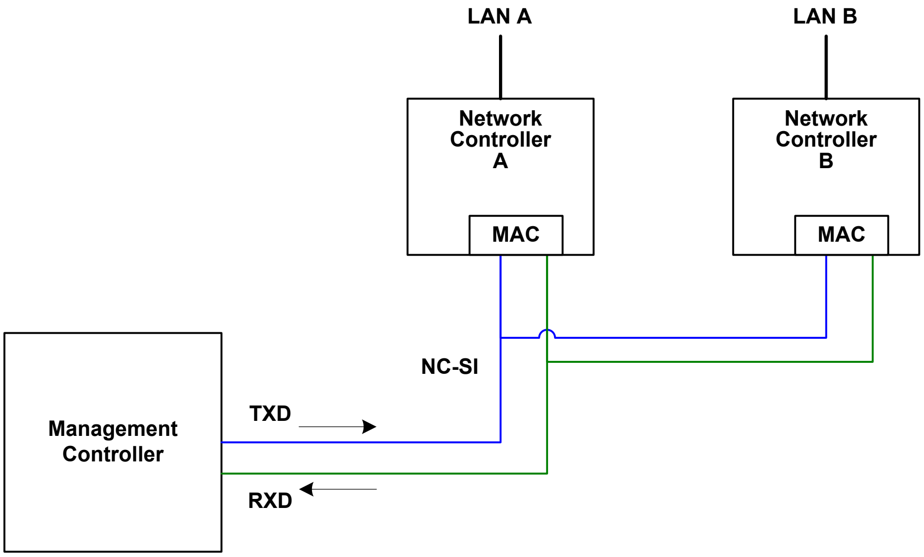


**Figure 10 – Basic multi-drop block diagram**

However, it is possible for multiple Network Controllers on the interface to be able to simultaneously *receive* traffic from the Management Controller that is being transmitted on the RBT TXD lines. The Network Controllers can receive commands from the Management Controller without having to arbitrate for the bus. This facilitates the Management Controller in delivering commands for setup and configuration of arbitration.

Arbitration allows multiple Network Controller packages that are attached to the interface to be enabled to share the RXD lines to deliver packets to the Management Controller.

This operation is summarized as follows:

- Only one Network Controller at a time can transmit packets on the RXD lines of the interface.
- Network Controllers can accept commands for configuring and controlling arbitration for the RXD lines.

## 7.3 Hardware arbitration

To prevent two or more NC-SI packages from transmitting at the same time, a hardware-based arbitration scheme was devised to allow only one Network Controller package to drive the RX lines of the shared interface at any given time. This scheme uses a mechanism of passing messages (opcodes) between Network Controller packages to coordinate when a controller is allowed to transmit through the RBT interface.

### 7.3.1 General

Three conceptual modes of hardware arbitration exist: arbitration master assignment, normal operation, and bypass. After a package is initialized and has its Channel IDs assigned, it enters the arbitration master assignment mode. This mode assigns one package the role of an Arbitration Master (ARB_Master) that is responsible for initially generating a TOKEN opcode that is required for the normal operating mode. In the normal operating mode, the TOKEN opcode is passed from one package to the next in the ring. The package is allowed to use the shared RXD signals and transmit if the package has received the TOKEN opcode and has a packet to send.

Bypass mode allows hardware arbitration opcodes to pass through a Network Controller package before it is initialized. Bypass mode shall be in effect while hardware arbitration is disabled. Bypass mode shall be exited, and arbitration master assignment mode shall be entered when the hardware arbitration becomes enabled or re-enabled.

Hardware-based arbitration requires two additional pins (ARB_IN and ARB_OUT*)* on the Network Controller. The ARB_OUT pin of one package is connected to the ARB_IN pin of the next package to form a ring configuration, as illustrated in Figure 11. The timing requirements for hardware arbitration are designed to accommodate a maximum of four Network Controller packages. If the implementation consists of a single Network Controller package, the ARB_OUT pin may be connected to the ARB_IN pin on the same package, or may be left disconnected, in which case hardware arbitration should be disabled by using the Select Package command. This specification optionally supports reporting of Hardware arbitration implementation status and hardware arbitration status using the **Get Capabilities** command.

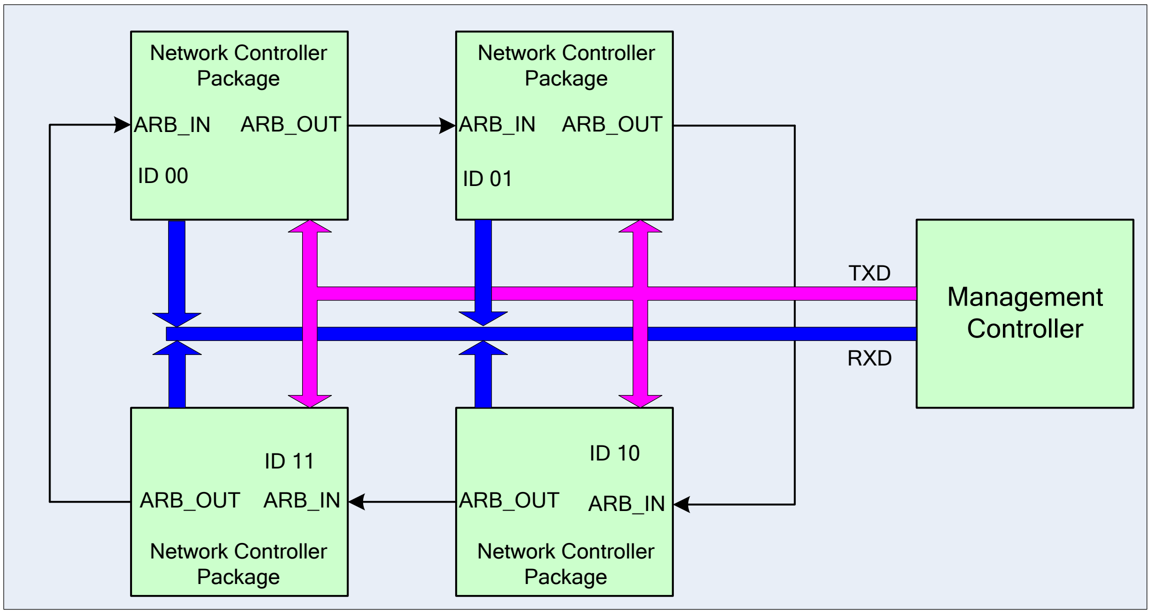

Network Controller Package
ARB_IN
ARB_OUT
ID 00
Network Controller Package
ARB_IN
ARB_OUT
ID 01
TXD
RXD
Management Controller
ID 11
ARB_OUT
ARB_IN
Network Controller Package
ID 10
ARB_OUT
ARB_IN
Network Controller Package

**Figure 11 – Multiple Network Controllers in a ring format**

Each Network Controller package sends out pulses on the ARB_OUT pin to create a series of symbols that form opcodes (commands) between Network Controllers. Each pulse is one clock wide and synchronized to REF_CLK. The hardware arbitration data bits follow the same timing specifications used for the TXD and RXD data bits (see clause 10.2.7). The pulses are di-bit encoded to ensure that symbols are correctly decoded. The symbols have the values shown in Table 5.

While clause 7.3.2.1 allows for opcode to be truncated, it is recommended that the transmission of current opcode on ARB_OUT be completed if the HW arbitration mode is changed in the middle of an opcode transfer (or in the middle of a symbol).

**Table 5 – Hardware arbitration di-bit encoding**

| Symbol Name | Encoded Value |
|---|---|
| $E_{sync}$ | 11b |
| $E_{zero}$ | 00b |
| $E_{one}$ | 01b |
| Illegal symbol | 10b |

### 7.3.2 Hardware arbitration opcodes

The hardware-based arbitration feature has five defined opcodes: IDLE, TOKEN, FLUSH, XON, and XOFF. Each opcode starts with an $E_{sync}$ symbol and is followed by either $E_{one}$ or $E_{zero}$ symbols. The legal opcodes are listed in Table 6.

**Table 6 – Hardware arbitration opcode format**

| Opcode | Format |
|---|---|
| IDLE | $E_{sync}$ $E_{zero}$ $E_{zero}$ (110000b) |
| TOKEN | $E_{sync}$ $E_{one}$ $E_{zero}$ (110100b) |
| FLUSH | $E_{sync}$ $E_{one}$ $E_{one}$ $E_{zero}$ E(Package_ID[2:0]) $E_{zero}$ (11010100xxxxxx00b) |
| XOFF | $E_{sync}$ $E_{zero}$ $E_{one}$ $E_{zero}$ $E_{zero}$ $E_{zero}$ (110001000000b) |
| XON | $E_{sync}$ $E_{zero}$ $E_{one}$ $E_{one}$ $E_{zero}$ E(Package_ID[2:0]) $E_{zero}$ (1100010100uuuuuu00b) |

#### 7.3.2.1 Detecting truncated opcodes

A truncated opcode is detected when the number of clocks between $E_{sync}$s is less than the number of bits required for the opcode. Note that any additional bits clocked in after a legitimate opcode is detected do not indicate an error condition and are ignored until the next $E_{sync}$.

#### 7.3.2.2 Handling truncated or illegal opcodes

When a Network Controller receives a truncated or illegal opcode, it should discard it.

#### 7.3.2.3 Relationship of opcodes processing and driving the RX data lines

A Network Controller package shall take no more than T9 REF_CLK times after receiving the last bit of the opcode to decode the incoming opcode and start generating the outgoing opcode. This time limit allows for decoding and processing of the incoming opcode under the condition that an outgoing opcode transmission is already in progress.

A package that has received a TOKEN and has packet data to transmit shall turn on its buffer and begin transmitting the packet data within T11 REF_CLK times of receiving the TOKEN, as illustrated in Figure 12. The package shall disable the RXD buffers before the last clock of the transmitted TOKEN.

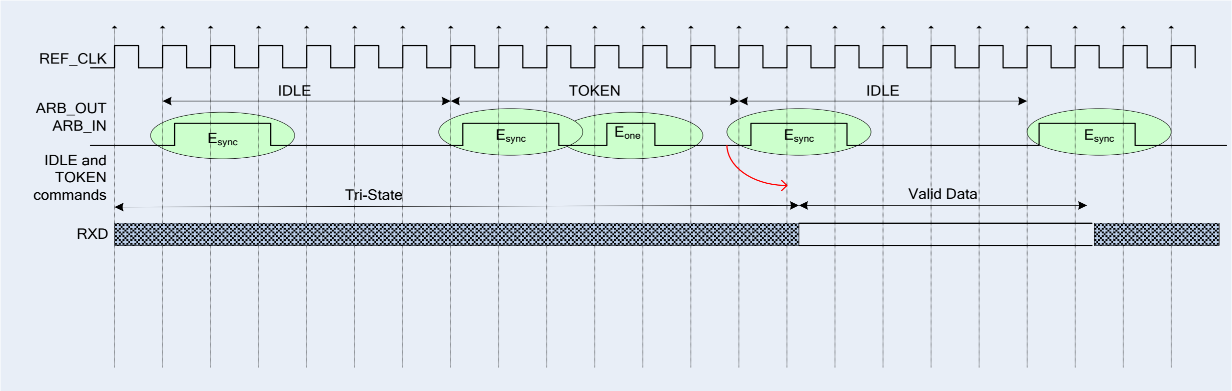


**Figure 12 – Opcode to RXD relationship**

### 7.3.3 Opcode operations

#### 7.3.3.1 TOKEN opcode

When a TOKEN opcode is received, the Network Controller package may drive the RXD signals to send only one of the following items: a Pass-through packet, a command response, or an AEN. One IEEE 802.3 PAUSE frame (XON or XOFF) may also be sent either before or after one of the previous packets, or on its own. While the Network Controller package is transmitting the data on the RXD signals of the interface, it shall generate IDLE opcodes on its ARB_OUT pin. Once a package completes its transmission, if any, it shall generate and send the TOKEN on its ARB_OUT pin.

#### 7.3.3.2 IDLE opcode

A package that has no other opcode to send shall continuously generate IDLE opcodes. Typically, a received IDLE opcode indicates that the TOKEN is currently at another package in the ring. This opcode is also used in the ARB_Master assignment process (for details, see clause 7.3.5). An Idle opcode typically will also be generated when the package is transmitting on RBT

#### 7.3.3.3 FLUSH opcode

A FLUSH opcode is used to establish an Arbitration Master for the ring when the package enters the Package Ready state or when the TOKEN is not received within the specified timeout, T8. This opcode is further explained in clause 7.3.5.

If the package receives a FLUSH opcode while it is in the middle of transmitting a packet onto NC-SI, it shall generate IDLE opcodes until the transmission is complete and then process the FLUSH opcode as described.

#### 7.3.3.4 Flow Control opcodes

The XON and XOFF opcodes are used to manage the generation of IEEE 802.3 PAUSE frames on the RBT interface. If the Network Controller supports flow control and flow control is enabled, the XOFF and XON opcodes behave as described in this clause. If the Network Controller does not support flow control or if flow control is not enabled, the Network Controller shall pass the opcodes to the next package.

There may be a configuration where some NCs support flow control and others do not. In this configuration, an NC sending an XOFF opcode may see the XOFF packet emission delayed by two or more full size Pass-through packets, one for each package not supporting XOFF when it gets the token, and one for the next package supporting XOFF before sending the XOFF packet. The NC is not required to provide buffering to prevent packet loss in this configuration. No drop behavior should be expected by an MC only if all NCs have flow control enabled.

NOTE: There is a maximum amount of time that the Network Controller is allowed to maintain a PAUSE. For more information, see clause 8.4.41.

##### 7.3.3.4.1 XOFF opcode

A Network Controller package that becomes congested while receiving packets from the NC-SI shall perform the following actions:

- If it does not have a TOKEN, it sends the XOFF opcode to the next package.

  NOTE: If it has the TOKEN and has not previously sent an XOFF frame for this instance of congestion, it shall send a single XOFF frame (PAUSE frame with a pause time of `0xFFFF`) and will not generate an XOFF opcode.

- A package may also regenerate an XOFF frame or opcode if it is still congested and determines that the present PAUSE frame is about to expire.

When a package on the ring receives an XOFF opcode, it shall perform one of the following actions:

- If it does not have a TOKEN opcode, it passes the XOFF opcode to the next package in the ring.
- If it has the TOKEN, it shall send an XOFF frame (PAUSE frame with a pause time of `0xFFFF`) and will not regenerate the XOFF opcode. If it receives another XOFF opcode while sending the XOFF frame or a regular network packet, it discards the received XOFF opcode.

#### 7.3.3.4.2 XON opcode

XON frames (PAUSE frame with a pause time of `0x0000`) are used to signal to the Management Controller that the Network Controller packages are no longer congested and that normal traffic flow can resume. XON opcodes are used between the packages to coordinate XON frame generation. The package ID is included in this opcode to provide a mechanism to verify that every package is not congested before sending an XON frame to the Management Controller.

The XON opcode behaves as follows:

- When a package is no longer congested, it generates an XON opcode with its own Package ID. This puts the package into the ‘waiting for its own XON’ state.
- A package that receives the XON opcode takes one of the following actions:
  - If it is congested, it replaces the received XON opcode with the IDLE opcode. This action causes the XON opcode to be discarded. Eventually, the congested package generates its own XON opcode when it exits the congested state.
  - If the package is not congested and is not waiting for the XON opcode with own Package ID, it forwards the received XON opcode to the next package in the ring.
  - If the received XON opcode contains the package’s own Package ID, the opcode should be discarded.
  - If the package is not congested and is waiting for its own XON opcode, it performs one of the following actions:
    - If it receives an XON opcode with a Package ID that is higher than its own, it replaces the XON opcode with its own Package ID.
    - If it receives an XON opcode with a Package ID lower than its own, it passes that XON opcode to the next package and it exits the ‘waiting for its own XON’ state.
    - If it receives an XON opcode with the Package ID equal to its own, it sends an XON frame on the NC-SI when it receives the TOKEN opcode and exits the ‘waiting for its own XON’ state.

      NOTE: More than one XON opcode with the same Package ID can be received while waiting for the TOKEN and while sending the XON frame. These additional XON opcodes should be discarded.
- If a package originates an XON opcode but receives an XOFF opcode, it terminates its XON request so that it does not output an XON frame when it receives the TOKEN.

  NOTE: This behavior is not likely to occur because the Management Controller will be in the Pause state at this point.
- A package that generated an XON opcode may receive its own XON opcode back while it has the TOKEN opcode. In this case, it may send a regular packet (Pass-through, command response, or AEN) to the Management Controller (if it has one to send), an XON frame, or both.

### 7.3.4 Bypass mode

When the Network Controller package is in bypass mode, data received on the ARB_IN pin is redirected to the ARB_OUT pin within the specified clock delay. This way, arbitration can continue between other devices in the ring.

A package in bypass mode shall take no more than T10 REF_CLK times to forward data from the ARB_IN pin to the ARB_OUT pin. The transition in and out of bypass mode may result in a truncated opcode.

A Network Controller package enters bypass mode immediately upon power up and transitions out of this mode after the Network Controller completes its startup/initialization sequence.

### 7.3.5 Hardware arbitration startup

Hardware arbitration startup works as follows:

1) All the packages shall be in bypass mode within Tpwrz seconds of NC-SI power up.

2) As each package is initialized, it shall continuously generate FLUSH opcodes with its own Package ID.

3) The package then participates in the ARB_MSTR assignment process described in the following clause.

### 7.3.6 ARB_MSTR assignment

ARB_MSTR assignment works as follows:

1) When a package receives a FLUSH opcode with a Package ID numerically smaller than its own, it shall forward on the received FLUSH opcode. If the received FLUSH opcode's Package ID is numerically larger than the local Package ID, the package shall continue to send its FLUSH opcode with its own Package ID. When a package receives a FLUSH opcode with its own Package ID, it becomes the master of the ring (ARB_MSTR).

2) The ARB_MSTR shall then send out IDLE opcodes until it receives an IDLE opcode.

3) Upon receiving the IDLE opcode, the ARB_MSTR shall be considered to be in possession of the TOKEN opcode (see clause 7.3.3.1).

4) If the package receives a FLUSH opcode while it is in the middle of transmitting a packet onto NC-SI, it shall generate IDLE opcodes until the transmission is complete and then process the FLUSH opcode as described.

### 7.3.7 Token timeout mechanism

Each Network Controller package that supports hardware-based arbitration control shall implement a timeout mechanism in case the TOKEN opcode is not received. When a package has a packet to send, it starts its timer. If it does not receive a TOKEN prior to the TOKEN timeout, the package shall send a FLUSH opcode. This restarts the arbitration process.

The timer may be programmable depending on the number of packages in the ring. The timeout value is designed to accommodate up to four packages, each sending the largest packet (1536 bytes) plus possible XON or XOFF frame transmission and opcode processing time. The timeout shall be no fewer than T8 cycles of the REF_CLK.

### 7.3.8 Timing considerations

The ARB_OUT and ARB_IN pins shall follow the timing specifications outlined in clause 10.

To improve the efficiency of the multi-drop NC-SI, TOKEN opcode generation may overlap the Inter Packet Gap (IPG) defined by the IEEE 802.3 specification, as shown in Figure 13. The TOKEN opcode shall be sent no earlier than the last T13 REF_CLK cycles of the IPG.

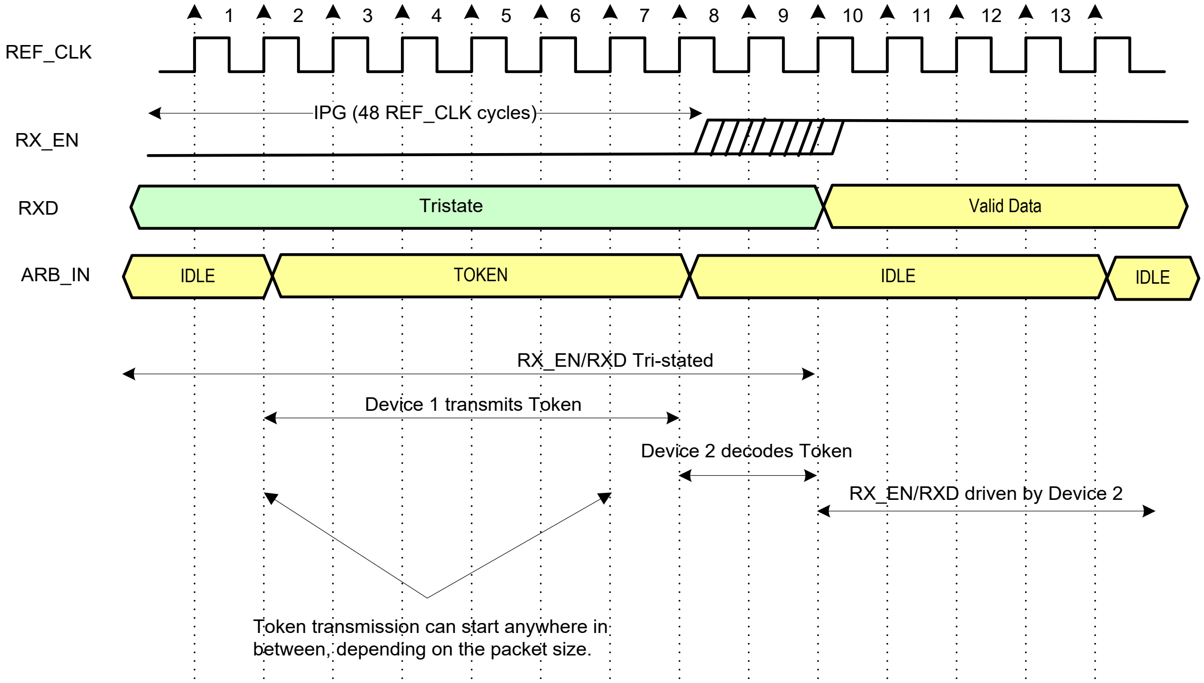


**Figure 13 – Example TOKEN to transmit relationship**

### 7.3.9 Example hardware arbitration state machine

The state machine diagram shown in Figure 14 is provided as a guideline to help illustrate the startup process and opcode operations described in the preceding clauses.

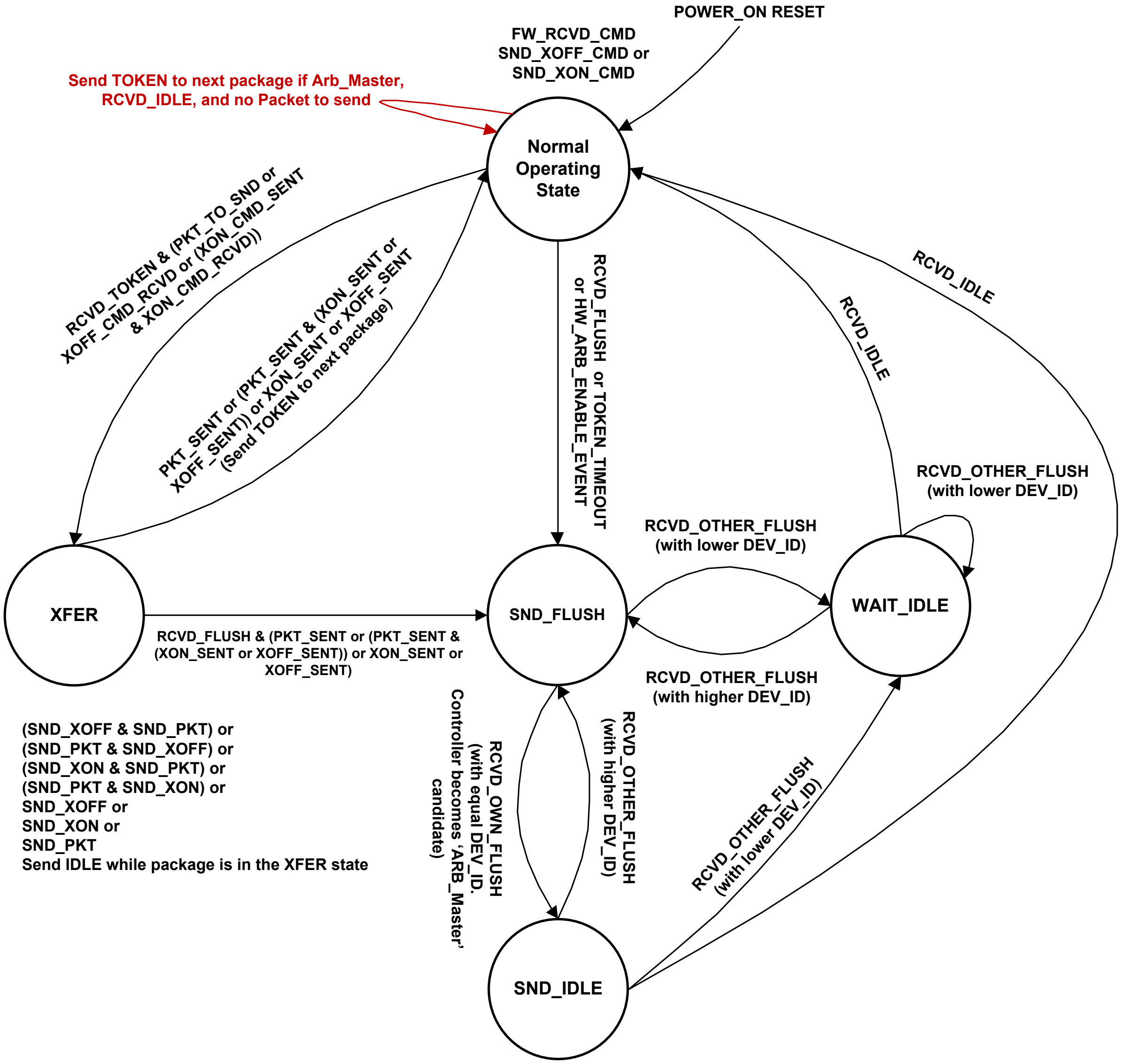


**Figure 14 – Hardware arbitration state machine**

The states and events shown in Figure 14 are described in Table 7 and Table 8, respectively.

**Table 7 – Hardware arbitration states**

| State | Action |
|---|---|
| Normal Operating State | This state is the normal operating state for hardware arbitration. The following actions happen in this state:<br>• FW_RCVD_CMD: Forward received command. As opcodes are received and acted upon, the resulting opcode is sent to the next package. For example, the TOKEN opcode is received, and no packet data is available to send, so the TOKEN opcode is sent to the next package in the ring.<br>• SND_XOFF_CMD: Send the XOFF opcode to the next package. This action happens when the specific conditions are met as described in clause 7.3.3.<br>• SND_XON_CMD: Send the XON opcode to the next package. This action happens when the specific conditions are met as described in clause 7.3.3.<br>• If the Network Controller is ARB_Master, it generates the TOKEN opcode upon receiving an IDLE opcode at the end of the FLUSH process.<br>• The RXD lines will be in a high-impedance condition in this state. |
| XFER | In this state, data is sent on the RXD lines. This data will be a Pass-through packet, response packet, XON (Pause Off) packet, XOFF (Pause On) packet, or AEN. (An XON or XOFF packet can be sent in addition to a Pass-through packet, response packet, or AEN.) IDLE opcodes are sent to the next package while the device is in the XFER state.<br>The following actions happen in this state:<br>• SND_XON: Transmit an XON frame (Pause Off) to the Management Controller.<br>• SND_XOFF: Transmit an XOFF frame (Pause On) to the Management Controller.<br>• SND_PKT: Transmit a Pass-through packet, response packet, or AEN to the Management Controller.<br>• The TOKEN opcode is sent to the next package upon completion of the transfer. |
| SND_FLUSH | This state is the entry point for determining the ARB_Master among the packages. In this state, the FLUSH opcode is continuously sent. This state is exited upon receiving a FLUSH opcode that has a DEV_ID that is equal to or lower than the package's own DEV_ID. |
| SND_IDLE | This is the final state for determining the ARB_Master, entered when a device's own FLUSH opcode is received. In this state, the IDLE opcode is continuously sent. |
| WAIT_IDLE | This state is entered when a FLUSH command is received from another package with a lower Device ID. When an IDLE opcode is received, the ARB_Master has been determined and the device transitions to the Normal Operating State. |

Table 8 – Hardware arbitration events

| Event | Description |
|---|---|
| RCVD_TOKEN | A TOKEN opcode was received, or the arbitration was just completed and won by this package. |
| RCVD_IDLE | An IDLE opcode was received. |
| XOFF_SENT | The Pause On frame was sent on the RXD interface. |
| XON_SENT | The Pause Off frame was sent on the RXD interface. |
| PKT_TO_SND | The Network Controller package has a Pass-through packet, command response packet, XON (Pause Off) frame, XOFF (Pause On) frame, or AEN to send. |
| XON_CMD_RCVD | A package received an XON opcode with its own Package ID. |
| XOFF_CMD_RCVD | An XOFF opcode was received. |
| XON_CMD_SENT | A package sent an XON opcode with its own Package ID. |
| RCVD_FLUSH | A FLUSH opcode was received. |
| TOKEN_TIMEOUT | The timeout limit expired while waiting for a TOKEN opcode. |
| HW_ARB_ENABLE_EVENT | This event begins ARB_MSTR assignment. This event occurs just after the Network Controller package initializes or when hardware arbitration is re-enabled through the Select Package command. |
| RCVD_OTHER_FLUSH | A package received a FLUSH opcode with a Package ID other than its own. |
| RCVD_OWN_FLUSH | A package received a FLUSH opcode with a Package ID equal to its own. |

## 7.4 Command-based arbitration

If hardware arbitration is not being used, the **Select Package** and **Deselect Package** commands shall be used to control which Network Controller package can transmit on the RXD lines. Because only one Network Controller package is allowed to transmit on the RXD lines, the Management Controller shall only have one package in the selected state at any given time. For more information, see clauses 8.4.5 and 8.4.7.

# 8 Packet definitions

## 8.1 NC-SI packet encapsulation

The RBT interface is an Ethernet interface adhering to the standard IEEE 802.3 Ethernet frame format. Whether or not the Network Controller accepts runt packets is unspecified.

As shown in Figure 15, this L2, or data link layer, frame format encapsulates all NC-SI packets, including Pass-through, command, and response packets, as the L2 frame payload data by adding a 14-byte header to the front of the data and appending a 4-byte Frame Check Sequence (FCS) to the end.

NC-SI Control Packets shall not include any VLAN tags. NC-SI Pass-through packets may include an 802.1Q VLAN tag.

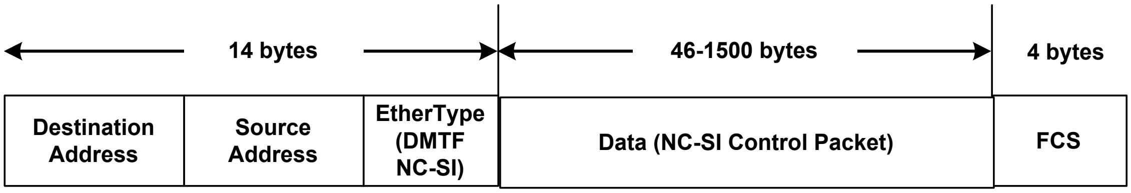


**Figure 15 – Ethernet frame encapsulation of NC-SI packet data without VLAN tag**

### 8.1.1 Ethernet frame header

The Management Controller shall format the 14-byte Ethernet frame header so that when it is received, it shall be formatted in the big-endian byte order shown in Table 9.

Channels shall accept Pass-through packets that meet the IEEE 802.3 frame requirements.

**Table 9 – Ethernet Header Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..03** | DA$_5$= 0xFF | DA$_4$= 0xFF | DA$_3$= 0xFF | DA$_2$= 0xFF |
| **04..07** | DA$_1$= 0xFF | DA$_0$= 0xFF | SA$_5$ | SA$_4$ |
| **08..11** | SA$_3$ | SA$_2$ | SA$_1$ | SA$_0$ |
| **12..13** | Ethertype = 0x88F8 (DMTF NC-SI) | | | |

#### 8.1.1.1 Destination Address (DA)

Bytes 0–5 of the header represent bytes 5–0 of the Ethernet Destination Address field of an L2 header.

The channel is not assigned a specific MAC address and the contents of this field are not interpreted as a MAC address by the Management Controller or the Network Controller. However, the DA field in all NC-SI Control Packets shall be set to the broadcast address (FF:FF:FF:FF:FF:FF) for consistency.

If the Network Controller receives a Control Packet with a Destination Address other than `FF:FF:FF:FF:FF:FF`, the Network Controller may elect to accept the packet, drop it, or return a response packet with an error response/reason code.

#### 8.1.1.2 Source Address (SA)

Bytes 6–11 of the header represent bytes 5–0 of the Ethernet Source MAC Address field of the Ethernet header. The contents of this field may be set to any value. The Network Controller should use `FF:FF:FF:FF:FF:FF` as the source address for NC-SI Control Packets that it generates.

#### 8.1.1.3 Ethertype

The final two bytes of the header, bytes 12..13, represent bytes 1..0 of the Ethertype field of the Ethernet header. For NC-SI Control Packets, this field shall be set to a fixed value of `0x88F8` as assigned to NC-SI by the IEEE. This value allows NC-SI Control Packets to be differentiated from other packets in the overall packet stream.

### 8.1.2 Frame Check Sequence

The Frame Check Sequence (FCS) shall be added at the end of the frame to provide detection of corruption of the frame. Any frame with an invalid FCS shall be discarded.

### 8.1.3 Data length

NC-SI Commands, Responses, and AENs do not carry any VLAN tag. NC-SI Commands, Responses and AENs shall have a payload data length between 46 and 1500 octets (bytes). This complies with the 802.3 specification. This means that the length of Ethernet frame shown in Figure 15 is between 64 octets (for a payload of 46 octets) and 1518 octets (for a payload with 1500 octets).

Pass-through packets also follow the 802.3 specification. The maximum payload size is 1500 octets; the minimum payload size shall be 42 octets when 802.1Q (VLAN) tag is present and 46 octets when the 802.1Q tag is not present. The Layer-2 Ethernet frame for an 802.1Q tagged frame shall be between 64 octets (for a payload of 42 octets) and 1522 octets (for a payload with 1500 octets). For Pass-through packets that are not 802.1Q tagged, the minimum Layer-2 Ethernet frame size is 64 octets (for a payload of 46 octets) and the maximum Layer-2 Ethernet frame size is 1518 octets (for a payload with 1500 octets).

## 8.2 Control Packet data structure

Each NC-SI Control Packet is made up of a 16-byte packet header and a payload section whose length is specific to the packet type.

### 8.2.1 Control Packet header

The 16-byte Control Packet header is used in command, response, and AEN packets, and contains data values intended to allow the packet to be identified, validated, and processed. The packet header is in big-endian byte order, as shown in Table 10.

**Table 10 – Control Packet header format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..03** | MC ID | Header Revision | Reserved | IID |
| **04..07** | Control Packet | Ch. ID | Flags | Payload Length |

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| | Type | | | |
| **08..11** | Reserved | | | |
| **12..15** | Reserved | | | |

#### 8.2.1.1 Management Controller ID

In Control Packets, this 1-byte field identifies the Management Controller issuing the packet. For this version of the specification, Management Controllers should set this field to `0x00` (zero). This implies that only one management controller is supported for accessing the NC via NC-SI at any given time, Network Controllers responding to command packets should copy the Management Controller ID field from the command packet header into the response packet header. For AEN packets, this field should be copied from the parameter that was set using the AEN Enable command.

#### 8.2.1.2 Header revision

This 1-byte field identifies the version of the Control Packet header in use by the sender. For this version of the specification, the header revision is `0x01`.

#### 8.2.1.3 Instance ID (IID)

This 1-byte field contains the IID of the command and associated response. The Network Controller can use it to differentiate retried commands from new instances of commands. The Management Controller can use this value to match a received response to the previously sent command. For more information, see clause 6.2.2.2.

#### 8.2.1.4 Control Packet type

This 1-byte field contains the Identifier that is used to identify specific commands and responses, and to differentiate AENs from responses. Each NC-SI command is assigned a unique 7-bit command type value in the range `0x00..0x60`. The proper response type for each command type is formed by setting the most significant bit (bit 7) in the original 1-byte command value. This allows for a one-to-one correspondence between 96 unique response types and 96 unique command types.

#### 8.2.1.5 Channel ID

This 1-byte field contains the Network Controller Channel Identifier. The Management Controller shall set this value to specify the package and internal channel ID for which the command is intended.

In a multi-drop configuration, all commands are received by all NC-SI Network Controllers present in the configuration. The Channel ID is used by each receiving Network Controller to determine if it is the intended recipient of the command. In Responses and AENs, this field carries the Channel ID l from which the response or AEN was issued.

#### 8.2.1.6 Payload length

This 12-bit field contains the length, in bytes, of any payload data present in the command or response frame following the NC-SI packet header. This value does not include the length of the NC-SI Control Packet Header, the checksum value, or any padding that might be present.

#### 8.2.1.7 Flags

Bit 0: Poll Indication: If this bit is set, it indicates that this command instance is polling on a previously sent command that was responded with a “Delayed Response” response code. This bit is relevant only for commands and not for responses or AENs.

Bits 3:1: Reserved

#### 8.2.1.8 Reserved

These fields are reserved for future use and should be written as zeros and ignored when read.

### 8.2.2 Control Packet payload

The NC-SI packet payload may contain zero or more defined data values depending on whether the packet is a command or response packet, and on the specific type. The NC-SI packet payload is always formatted in big-endian byte order, as shown in Table 11.

**Table 11 – Generic example of Control Packet payload**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..03** | Data0$_3$ | Data0$_2$ | Data0$_1$ | Data0$_0$ |
| **04..07** | Data1$_7$ | Data1$_6$ | Data1$_5$ | Data1$_4$ |
| **08..11** | Data1$_3$ | Data1$_2$ | Data1$_1$ | Data1$_0$ |
| **..** | | | | |
| **…** | DataN-1$_4$ | DataN-1$_3$ | DataN-1$_2$ | DataN-1$_1$ |
| **…** | DataN-1$_0$ | Payload Pad (as required) | | |
| **…** | Checksum | | | |
| **…** | Ethernet Packet Pad (as required) | | | |

#### 8.2.2.1 Data

As shown in Table 11, the bytes following the NC-SI packet header may contain payload data fields of varying sizes, and which may be aligned or require padding. In the case where data is defined in the payload, all data-field byte layouts (Data0–Data1) shall use big-endian byte ordering with the most significant byte of the field in the lowest addressed byte position (that is, coming first).

#### 8.2.2.2 Payload pad

If the payload is present and does not end on a 32-bit boundary, one to three padding bytes equal to `0x00` shall be present to align the checksum field to a 32-bit boundary.

#### 8.2.2.3 Checksum

This 4-byte field contains the 32-bit checksum compensation value that may be included in each command and response packet by the sender of the packet. When it is implemented, the checksum compensation shall be computed as the 2’s complement of the checksum, which shall be computed as the 32-bit unsigned sum of the NC-SI packet header and NC-SI packet payload interpreted as a series of 16-bit unsigned integer values. A packet receiver supporting packet checksum verification shall use the checksum compensation value to verify packet data integrity by computing the 32-bit checksum described above, adding to it the checksum compensation value from the packet, and verifying that the result is 0.

Verification of non-zero NC-SI packet checksum values is optional. An implementation may elect to generate the checksums and may elect to verify checksums that it receives. The checksum field is generated and handled according to the following rules:

- A checksum field value of all zeros specifies that a header checksum is not being provided for the NC-SI Control Packet, and that the checksum field value shall be ignored when processing the packet.
- If the originator of an NC-SI Control Packet is not generating a checksum, the originator shall use a value of all zeros for the header checksum field.
- If a non-zero checksum field is generated for an NC-SI Control Packet, that header checksum field value shall be calculated using the specified algorithm.
- All receivers of NC-SI Control Packets shall accept packets with all zeros as the checksum value (provided that other fields and the CRC are correct).
- The receiver of an NC-SI Control Packet may reject (silently discard) a packet that has an incorrect non-zero checksum.
- The receiver of an NC-SI Control Packet may ignore any non-zero checksums that it receives and accept the packet, even if the checksum value is incorrect (that is, an implementation is not required to verify the checksum field).
- A controller that generates checksums is not required to verify checksums that it receives.
- A controller that verifies checksums is not required to generate checksums for NC-SI Control Packets that it originates.

#### 8.2.2.4 Ethernet packet pad

Per IEEE 802.3, all Ethernet frames shall be at least 64 bytes in length, from the DA through and including FCS. For NC-SI packets, this requirement applies to the Ethernet header and payload, which includes the NC-SI Control Packet header and payload. Most NC-SI Control Packets are less than the minimum Ethernet frame payload size of 46 bytes in length and require padding to comply with IEEE 802.3.

### 8.2.3 Command packet payload

Command packets have no common fixed payload format.

### 8.2.4 Response packet payload

Unlike command packets that do not necessarily contain payload data, all response packets carry at least a 4-byte payload. This default payload carries the response codes and reason codes (described in clause 8.2.4.1) that provide status on the outcome of processing the originating command packet and is present in all response packet payload definitions.

The default payload occupies bytes `00..03` of the response packet payload, with any additional response-packet-specific payload defined to follow starting on the next word. All response packet payload fields are defined with big-endian byte ordering, as shown in Table 12.

**Table 12 – Generic example of Response packet payload format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..03** | Response Code | | Reason Code | |

| | | | | |
|---|---|---|---|---|
| **..** | ... | ... | ... | ... |
| **...** | DataN-$1_4$ | DataN-$1_3$ | DataN-$1_2$ | DataN-$1_1$ |
| **...** | DataN-$1_0$ | Word Pad (as required) | | |
| **...** | Checksum | | | |
| **...** | Ethernet Packet Pad (as required) | | | |

### 8.2.4.1 Response Packet in case of Delayed Response Code

If a response includes a “Delayed Response” Code, then the response does not contain the payload of the original response, The Delayed Response shall contain a payload of a single word (uint16) including the recommended next polling time in milliseconds. If no polling time estimate is available, then the recommended next polling time shall be set to `0x0000`.

**Table 13 – Generic example of Delayed Response packet payload**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..03** | Response Code = 0x0004 | | Reason Code = 0x0000 | |
| **04..07** | Reserved | | Next Polling time | |
| **08…11** | Checksum | | | |
| **…** | Ethernet Packet Pad (as required) | | | |

## 8.2.5 Response codes and reason codes

### 8.2.5.1 General

Response codes and reason codes are status values that are returned in the responses to NC-SI commands. The response code values provide a general categorization of the status being returned. The reason code values provide additional detail related to a particular response code.

Response codes and reason codes are divided into numeric ranges that distinguish whether the values represent standard codes that are defined in this specification or are vendor/OEM-specific values that are defined by the vendor of the controller.

The response code is a 2-byte field where values from `0x00` through `0x7F` are reserved for definition by this specification. Values from `0x80` through `0xFF` are vendor/OEM-specific codes that are defined by the vendor of the controller.

The reason code is a 2-byte field. The ranges of values are defined in Table 14.

**Table 14 – Reason code ranges**

| MS-byte | LS-byte | Description |
|---|---|---|
| `00h` | `0x00-0x7F` | Standard generic reason codes<br>This range of values for the lower byte is used for reason codes that are not specific to a particular command but can be used as reason codes in responses for any command. The values in this range are reserved for definition by this specification. |

| MS-byte | LS-byte | Description |
|---|---|---|
| | 0x80-0xFF | Vendor/OEM generic reason codes<br>This range of values for the lower byte is used for reason codes that are not specific to a particular command but can be used as reason codes in responses for any command. Values in this range are defined by the vendor of the controller. |
| Command Number<br>NOTE: This means that Command Number 00 cannot have any command-specific reason codes. | 0x00-0x7F | Standard command-specific reason codes<br>This range of values for the lower byte is used for reason codes that are specific to a particular command. The upper byte holds the value of the command for which the reason code is defined. The values in this range are reserved for definition by this specification. |
| | 0x80-0xFF | Vendor/OEM command-specific reason codes<br>This range of values for the lower byte is used for reason codes that are specific to a particular command. The upper byte holds the value of the command for which the reason code is defined. Values in this range are defined by the vendor of the controller. |

### 8.2.5.2 Response code and reason code values

The standard response code values are defined in Table 15, and the standard reason code values are defined in Table 16. Command-specific values, if any, are defined in the clauses that describe the response data for the command. Unless otherwise specified, the standard reason codes may be used in combination with any response code. There are scenarios where multiple combinations of response and reason code values are valid. Unless otherwise specified, an implementation may return any valid combination of response and reason code values for the condition.

**Table 15 – Standard response code values**

| Value | Description | Comment |
|---|---|---|
| 0x0000 | Command Completed | Returned for a successful command completion. When this response code is returned, the reason code shall be 0x0000 as described in Table 16 |
| 0x0001 | Command Failed | Returned to report that a valid command could not be processed or failed to complete correctly |
| 0x0002 | Command Unavailable | Returned to report that a command is temporarily unavailable for execution because the controller is in a transient state, busy condition, or in need of external intervention. |
| 0x0003 | Command Unsupported | Returned to report that a command is not supported by the implementation. The reason code “Unknown / Unsupported Command Type should be returned along with this response code for all unsupported commands. |
| 0x0004 | Delayed Response | Returned to report that the command was accepted, and the NC started to handle it, but it cannot respond within T5 seconds with a final answer.<br>When this response code is provided, the reason code shall be 0x0000. |
| 0x8000-0xFFFF | Vendor/OEM-specific | Response codes defined by the vendor of the controller |

Table 16 – Standard Reason Code Values

| Value | Description | Comment |
|---|---|---|
| `0x0000` | No Error/No Reason Code | When used with the Command Completed response code, indicates that the command completed normally. Otherwise this value indicates that no additional reason code information is being provided. |
| `0x0001` | Interface Initialization Required | Returned for all commands except Select/Deselect Package commands when the channel is in the Initial State, until the channel receives a Clear Initial State command |
| `0x0002` | Parameter Is Invalid, Unsupported, or Out-of-Range | Returned when a received parameter value is outside of the acceptable values for that parameter |
| `0x0003` | Channel Not Ready | Returned when the channel is in a transient state in which it is unable to process commands normally |
| `0x0004` | Package Not Ready | Returned when the package and channels within the package are in a transient state in which normal command processing cannot be done |
| `0x0005` | Invalid payload length | Returned when the payload length in the command is incorrect for the given command |
| `0x0006` | Information not available | Returned when the channel is unable to provide response data to a valid supported command. |
| `0x0007` | Intervention Required | May be returned for all commands, except for Select and Deselect Package, when the Package is not ready and requires intervention to restore its operational state.<br>When this code is returned, the NC does not check if the command is otherwise valid and the defined response is not returned. |
| `0x0008` | Link Command Failed-Hardware Access Error | Returned when PHY R/W access fails on Link commands |
| `0x0009` | Command Timeout | Command execution has exceeded the allocated T5 time |
| `0x000A` | Secondary Device Not Powered | A device that communicates with the NC is not powered up and cannot respond to the request |
| `0x000B-0x7FFE` | Reserved | |
| `0x7FFF` | Unknown / Unsupported Command Type | Returned when the command type is unknown or unsupported. This reason code shall only be used when the response code is 0x0003 (Command Unsupported) as described in Table 15. |
| `0x8000-0xFFFF` | OEM Reason Code | Vendor-specific reason code defined by the vendor of the controller |

### 8.2.6 AEN packet format

AEN packets shall follow the general packet format of Control Packets, with the IID field set to 0 because, by definition, the Management Controller does not send a response packet to acknowledge an AEN packet. The Control Packet Type field shall have the value `0xFF`. The originating Network Controller shall fill in the Channel ID (Ch. ID) field with its own ID to identify itself as the source of notification. The AEN Type field contains the identifier of what condition caused the generation of the AEN packet.

Table 17 represents the AEN packet format to be used for AENs defined in this specification.

**Table 17 – AEN packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..03** | MC ID = 0x0 | 0x01 | Reserved | IID = 0x0 |
| **04..07** | Control Packet Type = 0xFF | Originating Ch. ID | Reserved | Payload Length |
| **08..11** | Reserved | | | |
| **12..15** | Reserved | | | |
| **16..19** | Reserved | | | AEN Type |

Table 18 represents the AEN type ranges to be used for AENs defined in this specification.

**Table 18 – AEN Type Ranges**

| Value | AEN Type Allocation |
|---|---|
| 0x0..0x6F | Specification-defined AENs see clause 8.4.134; all others are Reserved |
| 0x70..0x7F | Transport-specific AENs |
| 0x80..0xFF | OEM-specific AENs |

### 8.2.7 Single OEM AEN packet format

OEM AEN packets shall conform to the format shown in Table 19 below for NCs that only support AENs using a single OEM identifier including NCs that implement spec version 1.1 and lower.

**Table 19 – OEM AEN packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..03** | MC ID = 0x0 | 0x01 | Reserved | IID = 0x0 |
| **04..07** | Control Packet Type = 0xFF | Originating Ch. ID | Reserved | Payload Length |
| **08..11** | Reserved | | | |
| **12..15** | Reserved | | | |
| **16..19** | Reserved | | | AEN Type |
| **20..23** | OPTIONAL AEN Data | | | |
| **24..27** | Checksum | | | |

### 8.2.8 Multiple OEMs AEN packet format

OEM AEN packets shall conform to the format shown in Table 20 below for NCs that support multiple OEM AENs and implement the Query and Set OEM AEN command.

**Table 20 – Multiple OEMs AEN packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..03** | MC ID = `0x0` | `0x01` | Reserved | IID = `0x0` |
| **04..07** | Control Packet Type = `0xFF` | Originating Ch. ID | Reserved | Payload Length |
| **08..11** | Reserved | | | |
| **12..15** | Reserved | | | |
| **16..19** | Reserved | | Multi field | AEN Type |
| | Manufacturer ID (IANA) | | | |
| **20..23** | OPTIONAL AEN Data | | | |
| **24..27** | Checksum | | | |

#### 8.2.8.1 Multi field

This field has a value of `0x01` to indicate the AEN contains a Manufacturer ID (IANA).

## 8.3 Control Packet type definitions

Command packet types are in the range of `0x00` to `0x7F`. Table 21 describes each command, its corresponding response, and the type value for each. Table 21 includes commands addressed to either a package or a channel. The commands addressed to a package are highlighted with gray background. PLDM and OEM-specific commands carried over NC-SI may be package specific or channel specific or both. The NC-SI over RBT binding specific commands are specifically called out in the table. The rest of the commands are general and apply to all NC-SI transport bindings.

Mandatory (M), Optional (O), and Conditional (C) refer to command support requirements for the Network Controller.

Ethernet (E), Fibre Channel (FC) and InfiniBand (IB) columns under the Fabric Implementation heading refer to the specific requirements of the NC implementing the network fabric type configured on the channel.

**Table 21 – Command and Response types**

| Control Packet Type | Command Name | Description | Response Packet Type | Fabric Implementation | | |
|---|---|---|---|---|---|---|
| | | | | E | FC | IB |
| `0x00` | Clear Initial State | Used by the Management Controller to acknowledge that the Network Controller is in the Initial State | `0x80` | M | M | M |
| `0x01` | Select Package | Used to explicitly select a controller package to transmit packets through the NC-SI interface | `0x81` | M | M | M |

| Control Packet Type | Command Name | Description | Response Packet Type | Fabric Implementation | | |
|---|---|---|---|---|---|---|
| | | | | E | FC | IB |
| 0x02 | Deselect Package | Used to explicitly instruct the controller package to stop transmitting packets through the NC-SI interface | 0x82 | M | M | M |
| 0x03 | Enable Channel | Used to enable the NC-SI channel and to cause the forwarding of bidirectional Management Controller packets to start | 0x83 | M | M | M |
| 0x04 | Disable Channel | Used to disable the NC-SI channel and to cause the forwarding of bidirectional Management Controller packets to cease | 0x84 | M | M | M |
| 0x05 | Reset Channel | Used to synchronously put the Network Controller back to the Initial State | 0x85 | M | M | M |
| 0x06 | Enable Channel Network TX | Used to explicitly enable the channel to transmit Pass-through packets onto the network | 0x86 | M | N/A | C |
| 0x07 | Disable Channel Network TX | Used to explicitly disable the channel from transmitting Pass-through packets onto the network | 0x87 | M | N/A | C |
| 0x08 | AEN Enable | Used to control generating AENs | 0x88 | C | C | C |
| 0x09 | Set Link | Used during OS absence to force link settings, or to return to auto-negotiation mode | 0x89 | M | N/A | C |
| 0x0A | Get Link Status | Used to get current link status information | 0x8A | M | N/A | C |
| 0x0B | Set VLAN Filter | Used to program VLAN IDs for VLAN filtering | 0x8B | M | N/A | C |
| 0x0C | Enable VLAN | Used to enable VLAN filtering of Management Controller RX packets | 0x8C | M | N/A | C |
| 0x0D | Disable VLAN | Used to disable VLAN filtering | 0x8D | M | N/A | C |
| 0x0E | Set MAC Address | Used to configure and enable unicast and multicast MAC address filters | 0x8E | M | N/A | C |

| Control Packet Type | Command Name | Description | Response Packet Type | Fabric Implementation | | |
|---|---|---|---|---|---|---|
| | | | | E | FC | IB |
| `0x10` | Enable Broadcast Filter | Used to enable selective broadcast packet filtering | `0x90` | M | N/A | C |
| `0x11` | Disable Broadcast Filter | Used to disable all broadcast packet filtering, and to enable the forwarding of all broadcast packets | `0x91` | M | N/A | C |
| `0x12` | Enable Global Multicast Filter | Used to enable selective multicast packet filtering | `0x92` | C | N/A | C |
| `0x13` | Disable Global Multicast Filter | Used to disable all multicast packet filtering, and to enable forwarding of all multicast packets | `0x93` | C | N/A | C |
| `0x14` | Set NC-SI Flow Control | Used to configure IEEE 802.3 flow control on RBT (RBT specific) | `0x94` | O | N/A | O |
| `0x15` | Get Version ID | Used to get controller-related version information | `0x95` | M | M | M |
| `0x16` | Get Capabilities | Used to get optional functions supported by the NC-SI | `0x96` | M | M | M |
| `0x17` | Get Parameters | Used to get configuration parameter values currently in effect on the controller | `0x97` | M | M | M |
| `0x18` | Get Controller Packet Statistics | Used to get current packet statistics for the Ethernet Controller | `0x98` | O | N/A | N/A |
| `0x19` | Get NC-SI Statistics | Used to request the packet statistics specific to the NC-SI | `0x99` | O | O | O |
| `0x1A` | Get NC-SI Pass-through Statistics | Used to request NC-SI Pass-through packet statistics | `0x9A` | O | N/A | O |
| `0x1B` | Get Package Status | Used to get current status of the package. | `0x9B` | O | O | O |
| | | | | | | |
| `0x25` | Get NC Capabilities and Settings | Used to request device configuration information and capabilities | `0xA5` | O | O | O |
| `0x26` | Set NC Configuration | Used to configure device interfaces | `0xA6` | O | O | O |
| `0x27` | Get PF | Used to request Function | `0xA7` | O | O | O |

| Control Packet Type | Command Name | Description | Response Packet Type | Fabric Implementation | | |
|---|---|---|---|---|---|---|
| | | | | E | FC | IB |
| | Assignment | assignment information | | | | |
| `0x28` | Set PF Assignment | Used to configure and enable Functions | `0xA8` | O | O | O |
| `0x29` | Get Channel Configuration | Used to request Channel configuration information | `0xA9` | O | O | O |
| `0x2A` | Set Channel Configuration | Used to configure operational characteristics of the Channel | `0xAA` | O | O | O |
| `0x2B` | Get Partition Configuration | Used to request partition configuration information | `0xAB` | O | O | O |
| `0x2C` | Set Partition Configuration | Used to configure partition operational characteristics | `0xAC` | O | O | O |
| `0x2D` | Get Boot Config | Used to request boot protocol configuration information | `0xAD` | O | O | O |
| `0x2E` | Set Boot Config | Used to configure boot protocol attributes | `0xAE` | O | O | O |
| `0x2F` | Get Partition Statistics | Used to request network link statistics for the partition | `0xAF` | O | O | O |
| `0x30` | Set Module Management Data | Used to configure management data of module | `0xB0` | O | O | O |
| `0x31` | Get FC Link Status | Used to request link and trunk status and speed for Fibre Channel ports | `0xB1` | N/A | M | N/A |
| `0x32` | Get Module Management Data | Used to retrieve management and inventory data of module | `0xB2` | O | O | O |
| `0x33` | Set Pass-through Mode Control | Used to enable/disable pass-through data paths of the NC | `0xB3` | O | O | O |
| `0x34` | Get Pass-through Mode Control | Used to retrieve states of pass-through data paths of the NC | `0xB4` | O | O | O |
| `0x35` | Get VF Allocation | Used to retrieve allocated VFs for PFs | `0xB5` | O | O | O |
| `0x36` | Set VF Allocation | Used to allocate VFs across PFs | `0xB6` | O | O | O |
| `0x38` | Get InfiniBand Link Status | Used to request link status for InfiniBand ports | `0xB8` | N/A | N/A | M |
| `0x39` | Get InfiniBand | Used to request port level | `0xB9` | N/A | N/A | M |

| Control Packet Type | Command Name | Description | Response Packet Type | Fabric Implementation | | |
|---|---|---|---|---|---|---|
| | | | | E | FC | IB |
| | Statistics | statistics for InfiniBand ports | | | | |
| 0x47 | Settings Commit | Used to request the commit of certain settings to NVRAM | 0xC7 | O | O | O |
| 0x48 | Get ASIC Temperature | Used to request current NC ASIC and other external device temperatures from the NC | 0xC8 | O | O | O |
| 0x49 | Get Ambient Temperature | Used to request the current ambient temperature from the NC adapter | 0xC9 | O | O | O |
| 0x4A | Get Transceiver Temperature | Used to request the current optical module temperature and thresholds | 0xCA | O | O | O |
| 0x4B | Thermal Shutdown Control | Used to control and query the state of the thermal-based self-shutdown feature | 0xCB | C | C | C |
| 0X4C | Transmit Data to NC | Used by the MC to transfer a block of data to the NC | 0xCC | O | O | O |
| 0X4D | Retrieve Data from NC | Used by the MC to transfer a block of data from the NC | 0xCD | O | O | O |
| 0x4E | Get Inventory Information | Used by the MC to get inventory data from the NC | 0xCE | O | O | O |
| | | | | | | |
| 0x50 | OEM Command | Used to request vendor-specific data | 0xD0 | O | O | O |
| 0x51-0x60 | Reserved for Transport Protocol Oriented Commands | Used to define transport protocol-oriented commands (e.g., PLDM over NC-SI/RBT) | 0xD1-0xE0 | O | O | O |
| | | | | | | |
| 0x51 | PLDM | Used by the MC to transfer a PLDM command (response) to (from) the NC (RBT specific) | 0xD1 | O | O | O |
| 0x52 | Get Package UUID | Returns a universally unique identifier (UUID) for the package<br>See MCTP DSP0261 for full definition<br>This command may be used on any transport | 0xD2 | O | O | O |

| Control Packet Type | Command Name | Description | Response Packet Type | Fabric Implementation | | |
|---|---|---|---|---|---|---|
| | | | | E | FC | IB |
| `0x54` | Get Supported Media | See MCTP DSP0261 for full definition<br>This command may be used on any transport | `0xD4` | O | O | O |
| `0x55` | Transport-specific AEN Enable | See MCTP DSP0261 for full definition | `0xD5` | O | O | O |
| `0x56` | Query Pending NC PLDM Request | Used by the MC to see if the NC has any pending PLDM requests to be retrieved (RBT specific) | `0xD6` | O | O | O |
| `0x57` | Send NC PLDM Reply | Used by the MC to provide a response to a previous SPDM request by the NC (RBT specific) | `0xD7` | O | O | O |
| `0x58` | Get MC MAC Address | Used by the MC to retrieve MAC addresses provisioned for its use<br>This command may be used on any transport | `0xD8` | O | N/A | O |
| | | | | | | |
| `0x60` | SPDM | Used by the MC to transfer a SPDM command (response) to (from) the NC (RBT specific) | `0xE0` | O | O | O |
| `0x61` | Query Pending NC SPDM Request | Used by the MC to see if the NC has any pending SPDM command to be retrieved (RBT specific) | `0xE1` | O | O | O |
| `0x62` | Send NC SPDM Reply | Used by the MC to respond to a previously read SPDM command from the NC (RBT specific) | `0xE2` | O | O | O |

## 8.4 Command and response packet formats

This clause describes the format for each of the NC-SI commands and corresponding responses.

The corresponding response packet format shall be mandatory when a given command is supported.

### 8.4.1 NC-SI command frame format

Table 22 illustrates the NC-SI frame format that shall be accepted by the Network Controller.

**Table 22 – Example of complete minimum-sized NC-SI command packet**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..03** | `0xFF` | `0xFF` | `0xFF` | `0xFF` |
| **04..07** | `0xFF` | `0xFF` | `0xXX` | `0xXX` |
| **08..11** | `0xXX` | `0xXX` | `0xXX` | `0xXX` |
| **12..15** | `0x88F8` | | MC ID | Header Revision |
| **16..19** | Reserved | IID | Command Type | Ch. ID |
| **20..23** | Reserved | Payload Length | Reserved | |
| **24..27** | Reserved | | Reserved | |
| **28..31** | Reserved | | Checksum (3..2) | |
| **32..35** | Checksum (1..0) | | Pad | |
| **36..39** | Pad | | | |
| **40..43** | Pad | | | |
| **44..47** | Pad | | | |
| **48..51** | Pad | | | |
| **52..55** | Pad | | | |
| **56..59** | Pad | | | |
| **60..63** | FCS | | | |

### 8.4.2 NC-SI response packet format

Table 23 illustrates the NC-SI response packet format that shall be transmitted by the Network Controller.

**Table 23 – Example of complete minimum-sized NC-SI response packet**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..03** | `0xFF` | `0xFF` | `0xFF` | `0xFF` |
| **04..07** | `0xFF` | `0xFF` | `0xFF` | `0xFF` |
| **08..11** | `0xFF` | `0xFF` | `0xFF` | `0xFF` |
| **12..15** | `0x88F8` | | MC ID | Header Revision |
| **16..19** | Reserved | IID | Response Type | Ch. ID |
| **20..23** | Reserved | Payload Length | Reserved | |
| **24..27** | Reserved | | Reserved | |
| **28..31** | Reserved | | Response Code | |
| **32..35** | Reason Code | | Checksum (3..2) | |
| **36..39** | Checksum (1..0) | | Pad | |
| **40..43** | Pad | | | |
| **44..47** | Pad | | | |

| | |
|---|---|
| 48..51 | Pad |
| 52..55 | Pad |
| 56..59 | Pad |
| 60..63 | FCS |

### 8.4.3 Clear Initial State command (0x00)

The Clear Initial State command provides the mechanism for the Management Controller to acknowledge that it considers a channel to be in the Initial State (typically because the Management Controller received an “Interface Initialization Required” reason code) and to direct the Network Controller to start accepting commands for initializing or recovering the NC-SI operation. When in the Initial State, the Network Controller shall return the “Interface Initialization Required” reason code for all channel commands until it receives the Clear Initial State command.

If the channel is in the Initial State when it receives the Clear Initial State command, the command shall cause the Network Controller to stop returning the “Interface Initialization Required” reason code. The channel shall also treat any subsequently received instance ID numbers as IDs for new command instances, not retries.

If the channel is not in the Initial State when it receives this command, it shall treat any subsequently received instance ID numbers as IDs for new command instances, not retries.

Table 24 illustrates the packet format of the Clear Initial State command.

**Table 24 – Clear Initial State command packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Control Packet Header | | | |
| 16..19 | Checksum | | | |
| 20..45 | Pad | | | |

### 8.4.4 Clear Initial State response (0x80)

Currently no command-specific reason code is identified for this response (see Table 25).

**Table 25 – Clear Initial State response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| 00..15 | NC-SI Control Packet Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Checksum | | | |
| 24..45 | Pad | | | |

### 8.4.5 Select Package command (`0x01`)

A package is considered to be “selected” when its NC-SI output buffers are allowed to transmit packets through the NC-SI interface. Conversely, a package is “deselected” when it is not allowed to transmit packets through the NC-SI interface.

The Select Package command provides a way for a Management Controller to explicitly take a package out of the deselected state and to control whether hardware arbitration is enabled for the package. (Similarly, the Deselect Package command allows a Management Controller to explicitly deselect a package.)

The NC-SI package in the Network Controller shall also become selected if the package receives any NC-SI command (other than Deselect Package) that is directed to the package or to a channel within the package.

The Select Package command is addressed to the package, rather than to a channel (that is, the command is sent with a Channel ID where the Package ID subfield matches the ID of the intended package and the Internal Channel ID subfield is set to `0x1F`).

More than one package can be in the selected state simultaneously if hardware arbitration is used between the selected packages and is active. The hardware arbitration logic ensures that buffer conflicts will not occur between selected packages.

If hardware arbitration is not active or is not used for a given package, only one package shall be selected at a time. To switch between packages, the Deselect Package command is used by the Management Controller to put the presently selected package into the deselected state before another package is selected.

A package shall stay in the selected state until it receives a Deselect Package command unless an internal condition causes all internal channels to enter the Initial State.

A package that is not using hardware arbitration may leave its output buffers enabled for the time that it is selected, or it may place its output buffers into the high-impedance state between transmitting packets through the NC-SI interface. (Temporarily placing the output buffers into the high-impedance state is not the same as entering the deselected state.)

For Type A integrated controllers: Because the RBT bus buffers are separately controlled, a separate Select Package command needs to be sent to each Package ID in the controller that is to be enabled to transmit through the NC-SI interface. If the internal packages do not support hardware arbitration, only one package shall be selected at a time; otherwise, a bus conflict will occur.

For Type S single channel, and Types B and C integrated controllers: A single set of RBT bus buffers exists for the package. Sending a Select Package command selects the entire package and enables all channels within the package to transmit through the NC-SI interface. (Whether a particular channel in a selected package starts transmitting Pass-through and AEN packets depends on whether that channel was enabled or disabled using the Enable or Disable Channel commands and whether the package may have had packets queued up for transmission.)

Implementation Note: The features control settings are only configurable via this command and are not altered by ‘implicit’ selection as described in clause 6.1.14.4.

Table 26 illustrates the packet format of the Select Package command.

Table 27 illustrates the disable byte for hardware arbitration.

**Table 26 – Select Package command packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Reserved | | | Features Control |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

**Table 27 – Features Control byte**

| Bits | Description |
|---|---|
| 0 | 0b = Hardware arbitration between packages is enabled.<br>1b = Disable hardware arbitration. Disabling hardware arbitration causes the package's arbitration logic to enter or remain in bypass mode.<br>In the case that the Network Controller does not support hardware arbitration, this bit is ignored; the Network Controller shall not return an error if the Select Package command can otherwise be successfully processed. |
| 1 | Delayed Response Enable:<br>0b = NC is not allowed to use the "Delayed Response" response code (default)<br>1b = NC is allowed to use the "Delayed Response" response code |
| 7..2 | Reserved |

### 8.4.6 Select Package response (0x81)

Currently no command-specific reason code is identified for this response (see Table 28).

**Table 28 – Select package response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.4.7 Deselect Package command (0x02)

The Deselect Package command directs the controller package to stop transmitting packets through the NC-SI interface and to place the output buffers for the package into the high-impedance state.

The Deselect Package command is addressed to the package, rather than to a particular channel (that is, the command is sent with a Channel ID where the Package ID subfield matches the ID of the intended package and the Internal Channel ID subfield is set to 0x1F).

The controller package enters the deselected state after it has transmitted the response to the Deselect Package command and placed its buffers into the high-impedance state. The controller shall place its outputs into the high-impedance state within the Package Deselect to Hi-Z Interval (T1). (This interval gives the controller being deselected time to turn off its electrical output buffers after sending the response to the Deselect Package command.)

If hardware arbitration is not supported or used, the Management Controller should wait for the Package Deselect to Hi-Z Interval (T1) to expire before selecting another controller.

For Type A integrated controllers: Because the bus buffers are separately controlled, putting the overall controller package into the high-impedance state requires sending separate Deselect Package commands to each Package ID in the overall package.

For Type S single channel, and Types B and C integrated controllers: A single set of bus buffers exists for the package. Sending a Deselect Package command deselects the entire NC-SI package and prevents all channels within the package from transmitting through the NC-SI interface.

Table 29 illustrates the packet format of the Deselect Package command.

**Table 29 – Deselect Package command packet format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

### 8.4.8 Deselect Package response (0x82)

The Network Controller shall always put the package into the deselected state after sending a Deselect Package Response.

No command-specific reason code is identified for this response (see Table 30).

**Table 30 – Deselect Package response packet format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.4.9 Enable Channel command (0x03)

The Enable Channel command shall enable the Network Controller to allow transmission of Pass-through and AEN packets to the Management Controller through the NC-SI.

Table 31 illustrates the packet format of the Enable Channel command.

**Table 31 – Enable Channel command packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

### 8.4.10 Enable Channel response (0x83)

No command-specific reason code is identified for this response (see Table 32).

**Table 32 – Enable Channel response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.4.11 Disable Channel command (0x04)

The Disable Channel command allows the Management Controller to disable the flow of packets, including Pass-through and AEN, to the Management Controller.

A Network Controller implementation is not required to flush pending packets from its RX Queues when a channel becomes disabled. If queuing is subsequently disabled for a channel, it is possible that a number of packets from the disabled channel could still be pending in the RX Queues. These packets may continue to be transmitted through the NC-SI interface until the RX Queues are emptied of those packets. The Management Controller should be aware that it may receive a number of packets from the channel before receiving the response to the Disable Channel command.

The 1-bit Allow Link Down (ALD) field can be used by the Management Controller to indicate that the link corresponding to the specified channel is not required after the channel is disabled. The Network Controller is allowed to take down the external network physical link if no other functionality (for example, host OS or WoL [Wake-on-LAN]) is active.

Possible values for the 1-bit ALD field are as follows:

- 0b = Keep link up (establish and/or keep a link established) while channel is disabled
- 1b = Allow link to be taken down while channel is disabled

Table 33 illustrates the packet format of the Disable Channel command.

**Table 33 – Disable Channel command packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Reserved | | | ALD |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

NOTE: It is currently unspecified whether this command will cause the Network Controller to cease the passing through of traffic from the Management Controller to the network, or if this can only be done using the Disable Channel Network TX command.

### 8.4.12 Disable Channel response (`0x84`)

No command-specific reason code is identified for this response (see Table 34).

**Table 34 – Disable Channel response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.4.13 Reset Channel command (`0x05`)

The Reset Channel command allows the Management Controller to put the channel into the Initial State. Packet transmission is not required to stop until the Reset Channel response has been sent. Thus, the Management Controller should be aware that it may receive a number of packets from the channel before receiving the response to the Reset Channel command.

Table 35 illustrates the packet format of the Reset Channel command.

**Table 35 – Reset Channel command packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Reserved | | | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.4.14 Reset Channel response (`0x85`)

Currently no command-specific reason code is identified for this response (see Table 36).

**Table 36 – Reset Channel response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.4.15 Enable Channel Network TX command (0x06)

The Enable Channel Network TX command shall enable the channel to transmit Pass-through packets onto the network. After network transmission is enabled, this setting shall remain enabled until a Disable Channel Network TX command is received, or the channel enters the Initial State.

The intention of this command is to control which Network Controller ports are allowed to transmit to the external network. The Network Controller compares the source MAC address in outgoing Pass-through packets to the unicast MAC address(es) configured using the Set MAC Address command. If a match exists, the packet is transmitted to the network.

Table 37 illustrates the packet format of the Enable Channel Network TX command.

**Table 37 – Enable Channel Network TX command packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

### 8.4.16 Enable Channel Network TX response (0x86)

No command-specific reason code is identified for this response (see Table 38).

**Table 38 – Enable Channel Network TX response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.4.17 Disable Channel Network TX command (0x07)

The Disable Channel Network TX command disables the channel from transmitting Pass-through packets onto the network. After network transmission is disabled, it shall remain disabled until an Enable Channel Network TX command is received.

Table 39 illustrates the packet format of the Disable Channel Network TX command.

**Table 39 – Disable Channel Network TX command packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Checksum | | | |
| **20..23** | Pad | | | |

### 8.4.18 Disable Channel Network TX response (0x87)

The NC-SI shall, in the absence of a checksum error or identifier mismatch, always accept the Disable Channel Network TX command and send a response.

Currently no command-specific reason code is identified for this response (see Table 40).

**Table 40 – Disable Channel Network TX response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.4.19 AEN Enable command (0x08)

Network Controller implementations shall support this command on the condition that the Network Controller generates one or more standard AENs. The AEN Enable command enables and disables the different standard AENs supported by the Network Controller. The Network Controller shall copy the AEN MC ID field from the AEN Enable command into the MC ID field in every subsequent AEN sent to the Management Controller.

The AEN Enable command is defined as both a package level command and a channel command. This means the command can be either addressed to the package (that is, the command is sent with the Internal Channel ID set to 0x1F) for configuring package level AENs or addressed to a specific channel in the package to configure AENs on that channel.

For more information on AEN, see clauses 8.5 (“AEN packet formats”) and 8.2.1.1 ("Management Controller ID").

Control of transport-specific AENs is outside the scope of this specification and should be defined by the transport binding specifications.

Table 41 illustrates the packet format of the AEN Enable command.

**Table 41 – AEN Enable command packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Reserved | | | AEN MC ID |
| **20..23** | AEN Control | | | |
| **24..27** | Checksum | | | |
| **28..45** | Pad | | | |

The AEN Control field has the format shown in Table 42.

**Table 42 – Format of AEN control**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | Link Status Change AEN control | 0b = Disable Link Status Change AEN<br>1b = Enable Link Status Change AEN |
| 1 | Configuration Required AEN control | 0b = Disable Configuration Required AEN<br>1b = Enable Configuration Required AEN |
| 2 | Host NC Driver Status Change AEN control | 0b = Disable Host NC Driver Status Change AEN<br>1b = Enable Host NC Driver Status Change AEN |
| 3 | Delayed Response Ready AEN control | 0b = Disable Delayed Response Ready AEN<br>1b = Enable Delayed Response Ready AEN |
| 4 | InfiniBand Link Status Change AEN control | 0b = Disable IB Link Status Change AEN<br>1b = Enable IB Link Status Change AEN |
| 5 | Fibre Channel Link Status Change AEN control | 0b = Disable FC Link Status Change AEN<br>1b = Enable FC Link Status Change AEN |
| 6 | Transceiver Event AEN Control | 0b = Disable Transceiver Event AEN<br>1b = Enable Transceiver Event AEN |
| 7 | Request Data Transfer AEN control | 0b = Disable Request Data Transfer AEN<br>1b = Enable Request Data Transfer AEN |
| 8 | Partition Link Status Change AEN control | 0b = Disable Partition Link Status Change AEN<br>1b = Enable Partition Link Status Change AEN |
| 9 | Thermal Shutdown Event AEN control | 0b = Disable Thermal Shutdown Event AEN<br>1b = Enable Thermal Shutdown Event AEN |
| 15..10 | Reserved | Reserved |
| 31..16 | OEM-specific AEN control | OEM-specific control |

### 8.4.20 AEN Enable response (`0x88`)

Currently no command-specific reason code is identified for this response (see Table 43). If the MC attempts to set an AEN type that is not supported, the NC shall reject the entire command even if it also includes valid AENs and respond with the “Command Failed” response and “Parameter Is Invalid…” reason codes.

**Table 43 – AEN Enable response packet format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.4.21 Set Link command (`0x09`)

The Set Link command may be used by the Management Controller to configure the external network interface associated with the channel by using the provided settings. Upon receiving this command, while the host NC driver is not operational, the channel shall attempt to set the link to the configuration specified by the parameters. Upon successful completion of this command, link settings specified in the command should be used by the network controller as long as the host NC driver does not overwrite the link settings.

In the absence of an operational host NC driver, the NC should attempt to make the requested link state change even if it requires the NC to drop the current link. The channel shall send a response packet to the Management Controller within the required response time. However, this specification does not specify the amount of time the requested link state changes may take to complete.

The actual link settings are controlled by the host NC driver when it is operational. When the host NC driver is operational, link settings specified by the MC using the Set Link command may be overwritten by the host NC driver. The link settings are not restored by the NC if the host NC driver becomes non-operational.

Table 44 illustrates the packet format of the Set Link command.

**Table 44 – Set Link command packet format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Link Settings | | | |
| **20..23** | OEM Link Settings | | | |
| **24..27** | Checksum | | | |
| **28..45** | Pad | | | |

Table 45 and Table 46 describe the Set Link bit definitions. Refer to IEEE 802.3 for definitions of Auto Negotiation, Duplex Setting, Pause Capability, and Asymmetric Pause Capability. The Error correction, Auto Negotiation, Duplex, Modulation Scheme, Parallel Detect, Pause Capability, Asymmetric Pause Capability, and Energy Efficient Ethernet bits shall be ignored for the link that is not Ethernet link.

**Table 45 – Set Link bit definitions**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 00 | Auto Negotiation<br>If Auto Negotiation is not used, only one combination of single link speed, protocol and FEC settings is allowed to be configured, otherwise a Command Failed response code and Parameter Is Invalid, Unsupported, or Out-of-Range reason code shall be returned. | 1b = enable<br>0b = disable |
| 01..07 | Link Speed Selection<br>More than one speed can be selected when Auto Negotiation is set to 'enable'. If Auto Negotiation is not used, the channel attempts to force the link to the specified setting (in this case, if the setting is not supported or if multiple speeds are enabled, a Command Failed response code and Parameter Is Invalid, Unsupported, or Out-of-Range reason code shall be returned). . If multiple settings are enabled, a Command Failed response code and Set Link Speed Conflict reason code shall be returned)<br>NOTE Additional link speeds are defined below. | Bit 01: 1b = enable 10 Mbps<br>Bit 02: 1b = enable 100 Mbps<br>Bit 03: 1b = enable 1000 Mbps (1 Gbps)<br>Bit 04: 1b = enable 10 Gbps<br>Bit 05: 1b = enable 20 Gbps (optional for NC-SI 1.1, Reserved for NC-SI 1.0)<br>Bit 06: 1b = enable 25 Gbps (optional for NC-SI 1.1, Reserved for NC-SI 1.0)<br>Bit 07: 1b = enable 40 Gbps (optional for NC-SI 1.1, Reserved for NC-SI 1.0) |
| 08..09 | Duplex Setting<br>(separate duplex setting bits)<br>More than one duplex setting can be selected when Auto Negotiation is set to 'enable'. If Auto Negotiation is not used, the channel attempts to force the link to the specified setting (in this case, if the setting is not supported or if multiple settings are enabled, a Command Failed response code and Parameter Is Invalid, Unsupported, or Out-of-Range reason code shall be returned." | Bit 08: 1b = enable half-duplex<br>Bit 09: 1b = enable full-duplex |
| 10 | Pause Capability<br>If Auto Negotiation is not used, the channel should apply pause settings assuming the partner supports the same capability. | 1b = disable<br>0b = enable |
| 11 | Asymmetric Pause Capability<br>If Auto Negotiation is not used, the channel should apply asymmetric pause settings assuming the partner supports the same capability. | 1b = enable<br>0b = disable |
| 12 | OEM Link Settings Field Valid (see Table 46) | 1b = enable<br>0b = disable |
| 13..19 | Additional Link Speeds (see Link Speed Selection) | Bit 13: 1b = enable 50 Gbps (optional for NC-SI 1.1, Reserved for NC-SI 1.0)<br>Bit 14: 1b = enable 100 Gbps (optional for NC-SI 1.1, Reserved for NC-SI 1.0)<br>Bit 15: 1b = enable 2.5 Gbps (optional for NC-SI 1.1, Reserved for NC-SI 1.0) |

| Bit Position | Field Description | Value Description |
|---|---|---|
| | | Bit 16: 1b = enable 5 Gbps (optional for NC-SI 1.1, Reserved for NC-SI 1.0)<br>Bit 17: 1b = enable 200 Gbps (optional for NC-SI 1.2, Reserved for NC-SI 1.1/1.0)<br>Bit 18: 1b = enable 400 Gbps (optional for NC-SI 1.2, Reserved for NC-SI 1.1/1.0)<br>Bit 19: 1b = enable 800 Gbps (optional for NC-SI 1.2, Reserved for NC-SI 1.1/1.0) |
| 20..21 | Reserved | |
| 22..23 | Modulation Scheme | Bit 22: 1b = NRZ (optional for NC-SI 1.2, Reserved for NC-SI 1.1/1.0)<br>Bit 23: 1b = PAM-4 (optional for NC-SI 1.2, Reserved for NC-SI 1.1/1.0)<br>Bit 23-22 Values:<br>00 – Use default<br>01 – Enable NRZ<br>10 – Enable PAM-4<br>11 – Enable NRZ and PAM-4 |
| 24..27 | Forward Error Correction (FEC) Algorithm | Bit 24: 1b = BASE-R FEC (Firecode) (optional for NC-SI 1.2, Reserved for NC-SI 1.1/1.0)<br>Bit 25: 1b = RS-FEC (Reed Solomon) (optional for NC-SI 1.2, Reserved for NC-SI 1.1/1.0)<br>Bit 26..27 Reserved<br>If all bits are set to 0, then no FEC algorithm shall be selected |
| 28 | Energy Efficient Ethernet (EEE) | 1b = enable<br>0b = disable |
| 29 | Link Training (LT) | 1b = enable<br>0b = disable |
| 30 | Parallel Detect<br>An auto-negotiation link partner's mechanism to establish links with non-negotiation, fixed-speed linked partners. | 1b = enable<br>0b = disable |
| 31 | Reserved | 0 |

2444 **Table 46 – OEM Set Link bit definitions**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 00..31 | OEM Link Settings | Vendor specified |

### 8.4.22 Set Link Response (0x89)

The channel shall, in the absence of a checksum error or identifier mismatch, always accept the Set Link command and send a response (see Table 47). In the presence of an operational Host NC driver, the NC should not attempt to make link state changes and should send a response with reason code 0x1 (Set Link Host OS/ Driver Conflict).

If the Auto Negotiation field is set, the NC should ignore Link Speed Selection and Duplex Setting fields that are not supported by the NC.

**Table 47 – Set Link response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

Table 48 describes the reason codes that are specific to the Set Link command. Returning the following command-specific codes is recommended, conditional upon Network Controller support for the related capabilities.

**Table 48 – Set Link command-specific reason codes**

| Value | Description | Comment |
|---|---|---|
| 0x0901 | Set Link Host OS/ Driver Conflict | Returned when the Set Link command is received when the Host NC driver is operational |
| 0x0902 | Set Link Media Conflict | Returned when Set Link command parameters conflict with the media type (for example, Fiber Media) |
| 0x0903 | Set Link Parameter Conflict | Returned when Set Link parameters conflict with each other (for example, 1000 Mbps HD with copper media) |
| 0x0904 | Set Link Power Mode Conflict | Returned when Set Link parameters conflict with current low-power levels by exceeding capability |
| 0x0905 | Set Link Speed Conflict | Returned when Set Link parameters attempt to force more than one speed at the same time when Auto Negotiation is disabled |
| 0x0906 | Link Command Failed-Hardware Access Error | Returned when PHY R/W access fails to complete normally while executing the Set Link or Get Link Status command |
| 0x0907 | Set Link SerDes Conflict | Returned when Set Link parameters attempt to force an unsupported SerDes configuration |
| 0x0908 | Set Link FEC Conflict | Returned when Set Link parameters attempt to force an unsupported FEC algorithm |
| 0x0909 | Set Link EEE Conflict | Returned when Set Link parameters attempt to force an unsupported EEE configuration |

| Value | Description | Comment |
|---|---|---|
| `0x090A` | Set Link LT Conflict | Returned when Set Link parameters attempt to force an unsupported link training configuration |
| `0x090B` | Set Link Parallel Detection Conflict | Returned when Set Link parameters attempt to force an unsupported parallel detection configuration |

### 8.4.23 Get Link Status command (`0x0A`)

The Get Link Status command allows the Management Controller to query the channel for potential link status and error conditions (see Table 49).

**Table 49 – Get Link Status command packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

### 8.4.24 Get Link Status response (`0x8A`)

The channel shall, in the absence of a checksum error or identifier mismatch, always accept the Get Link Status command and send a response (see Table 50).

**Table 50 – Get Link Status response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Link Status | | | |
| **24..27** | Other Indications | | | |
| **28..31** | OEM Link Status | | | |
| **32..35** | Checksum | | | |
| **36..45** | Pad | | | |

Table 51 describes the Link Status bit definitions.

Table 51 – Link Status field bit definitions

| Bit Position | Field Description | Value Description |
|---|---|---|
| 00 | Link Flag | 0b = Link is down<br>1b = Link is up (including Low Power Idle state in EEE)<br>This field is mandatory. |
| 04..01 | Speed and duplex | 0x0 = Auto-negotiate not complete [per IEEE 802.3], or SerDes Flag = 1b, or no Highest Common Denominator (HCD) from the following options (0x1 through 0xF) was found.<br>0x1 = 10BASE-T half-duplex<br>0x2 = 10BASE-T full-duplex<br>0x3 = 100BASE-TX half-duplex<br>0x4 = 100BASE-T4<br>0x5 = 100BASE-TX full-duplex<br>0x6 = 1000BASE-T half-duplex<br>0x7 = 1000BASE-T full-duplex<br>0x8 = 10G-BASE-T support or 10 Gbps<br>0x9 = 20 Gbps (optional for NC-SI 1.1, Reserved for NC-SI 1.0)<br>0xA = 25 Gbps (optional for NC-SI 1.1, Reserved for NC-SI 1.0)<br>0xB = 40 Gbps (optional for NC-SI 1.1, Reserved for NC-SI 1.0)<br>0xC = 50 Gbps (optional for NC-SI 1.1, Reserved for NC-SI 1.0)<br>0xD = 100 Gbps (optional for NC-SI 1.1, Reserved for NC-SI 1.0)<br>0xE = 2.5 Gbps (optional for NC-SI 1.1, Reserved for NC-SI 1.0)<br>0xF = Use values defined in Extended Speed and Duplex field starting at bit 24 (optional for NC-SI 1.1, Reserved for NC-SI 1.0)<br>When SerDes Flag = 0b, the value may reflect forced link setting.<br>NOTE For physical medium and/or speed/duplex not listed above, the closest speed and duplex option can be reported by the NC. This field does not infer any media type information. |
| 05 | Auto Negotiate Flag | 1b = Auto-negotiation is enabled.<br>This field always returns 0b if auto-negotiation is not supported or is not enabled or when the link is not an Ethernet link.<br>This field is mandatory if supported by the controller. |
| 06 | Auto Negotiate Complete | 1b = Auto-negotiation has completed.<br>This includes if auto-negotiation was completed using Parallel Detection. Always returns 0b if auto-negotiation is not supported or is not enabled or when the link is not an Ethernet link.<br>This field is mandatory if the Auto Negotiate Flag is supported. |
| 07 | Parallel Detection Flag | 1b = Link partner did not support auto-negotiation and parallel detection was used to get link.<br>This field contains 0b if Parallel Detection was not used to obtain link or when the link is not an Ethernet link. |
| 08 | Reserved | None |

| Bit Position | Field Description | Value Description |
|---|---|---|
| 09 | Link Partner Advertised Speed and Duplex 1000TFD | 1b = Link Partner is 1000BASE-T full-duplex capable.<br>Valid only for Ethernet Link and when:<br>SerDes Flag = 0b<br>Auto-Negotiate Flag = 1b<br>Auto-Negotiate Complete = 1b<br>This field is mandatory. |
| 10 | Link Partner Advertised Speed and Duplex 1000THD | 1b = Link Partner is 1000BASE-T half-duplex capable.<br>Valid only for Ethernet Link and when:<br>SerDes Flag = 0b<br>Auto-Negotiate Flag = 1b<br>Auto-Negotiate Complete = 1b<br>This field is mandatory. |
| 11 | Link Partner Advertised Speed 100T4 | 1b = Link Partner is 100BASE-T4 capable.<br>Valid only for Ethernet Link and when:<br>SerDes Flag = 0b<br>Auto-Negotiate Flag = 1b<br>Auto-Negotiate Complete = 1b<br>This field is mandatory. |
| 12 | Link Partner Advertised Speed and Duplex 100TXFD | 1b = Link Partner is 100BASE-TX full-duplex capable.<br>Valid only for Ethernet Link and when:<br>SerDes Flag = 0b<br>Auto-Negotiate Flag = 1b<br>Auto-Negotiate Complete = 1b<br>This field is mandatory. |
| 13 | Link Partner Advertised Speed and Duplex 100TXHD | 1b = Link Partner is 100BASE-TX half-duplex capable.<br>Valid only for Ethernet Link and when:<br>SerDes Flag = 0b<br>Auto-Negotiate Flag = 1b<br>Auto-Negotiate Complete = 1b<br>This field is mandatory. |
| 14 | Link Partner Advertised Speed and Duplex 10TFD | 1b = Link Partner is 10BASE-T full-duplex capable.<br>Valid only for Ethernet Link and when:<br>SerDes Flag = 0b<br>Auto-Negotiate Flag = 1b<br>Auto-Negotiate Complete = 1b<br>This field is mandatory. |

| Bit Position | Field Description | Value Description |
|---|---|---|
| 15 | Link Partner Advertised Speed and Duplex 10THD | `1b` = Link Partner is 10BASE-T half-duplex capable.<br>Valid only for Ethernet Link and when:<br>SerDes Flag = `0b`<br>Auto-Negotiate Flag = `1b`<br>Auto-Negotiate Complete = `1b`<br>This field is mandatory. |
| 16 | TX Flow Control Flag | `0b` = Transmission of Pause frames by the NC onto the external network interface is disabled or this is non-Ethernet link.<br>`1b` = Transmission of Pause frames by the NC onto the external network interface is enabled.<br>This field is mandatory. |
| 17 | RX Flow Control Flag | `0b` = Reception of Pause frames by the NC from the external network interface is disabled or this is non-Ethernet link.<br>`1b` = Reception of Pause frames by the NC from the external network interface is enabled.<br>This field is mandatory. |
| 19..18 | Link Partner Advertised Flow Control | `00b` = Link partner is not pause capable or this is non-Ethernet link.<br>`01b` = Link partner supports symmetric pause.<br>`10b` = Link partner supports asymmetric pause toward link partner.<br>`11b` = Link partner supports both symmetric and asymmetric pause.<br>Valid when:<br>SerDes Flag = `0b`<br>Auto-Negotiate = `1b`<br>Auto-Negotiate Complete = `1b`<br>This field is mandatory. |
| 20 | SerDes Link | SerDes status (See 4.22.)<br>`0b` = SerDes is not used or used to connect to an external PHY<br>`1b` = SerDes is used as a direct attach interface<br>This field is mandatory. |
| 21 | OEM Link Speed Valid | `0b` = OEM link settings are invalid.<br>`1b` = OEM link settings are valid. |
| 23..22 | Modulation Scheme | `00b` = Reserved or this is non-Ethernet link<br>`01b` = NRZ is used.<br>`10b` = PAM-4 is used.<br>`11b` = Reserved<br>NOTE: This field is optional for NC-SI 1.2, reserved for NC-SI 1.1/1.0. |

| Bit Position | Field Description | Value Description |
|---|---|---|
| 31..24 | Extended Speed and duplex | Optional for NC-SI 1.2/1.1, Reserved for NC-SI 1.0<br>0x0 = Auto-negotiation not complete [per IEEE 802.3], or<br>SerDes Flag = 1b, or<br>no highest common denominator speed from the following options (0x01 through 0x0F) was found.<br>0x01 = 10BASE-T half-duplex<br>0x02 = 10BASE-T full-duplex<br>0x03 = 100BASE-TX half-duplex<br>0x04 = 100BASE-T4<br>0x05 = 100BASE-TX full-duplex<br>0x06 = 1000BASE-T half-duplex<br>0x07 = 1000BASE-T full-duplex<br>0x08 = 10G-BASE-T support or 10 Gbps<br>0x09 = 20 Gbps<br>0x0A = 25 Gbps<br>0x0B = 40 Gbps<br>0x0C = 50 Gbps<br>0x0D = 100 Gbps<br>0x0E = 2.5 Gbps<br>0x0F = 5 Gbps<br>0x10 = 1 Gbps (for non Base-T)<br>0x11 = 200 Gbps<br>0x12 = 400 Gbps<br>0x13 = 800 Gbps<br>0x14-0xFF = Reserved<br>When SerDes Flag = 0b, the value may reflect forced link setting.<br>NOTE: For the physical medium and/or speed/duplex not listed above, the closest speed and duplex option can be reported by the NC. This field does not infer any media type information. |

Table 52 describes the Other Indications field bit definitions.

**Table 52 – Other Indications field bit definitions**

| Bits | Description | Values |
|---|---|---|
| 00 | Host NC Driver Status Indication | 0b = The Network Controller driver for the host external network interface associated with this channel is not operational (not running), unknown, or not supported.<br>1b = The Network Controller driver for the host external network interface associated with this channel (or when partitioned, at least one partition driver) is being reported as operational (running).<br>This bit always returns 0b if the Host NC Driver Status Indication is not supported. |
| 01 | Energy Efficient Ethernet (EEE) | 1b = enabled<br>0b = disabled or this is non-Ethernet link |

| Bits | Description | Values |
|---|---|---|
| 02 | Link Training (LT) | 1b = enabled<br>0b = disabled or this is non-Ethernet link |
| 03 | Parallel Detect | 1b = enabled<br>0b = disabled or this is non-Ethernet link |
| 04 | OEM Link Status Field | 1b = enabled<br>0b = disabled |
| 05..31 | Reserved | |

Table 53 describes the OEM Link Status field bit definitions.

**Table 53 – OEM Link Status field bit definitions (optional)**

| Bits | Description | Values |
|---|---|---|
| 00..31 | OEM Link Status | OEM specific |

Table 54 describes the reason code that is specific to the Get Link Status command.

**Table 54 – Get Link Status command-specific reason code**

| Value | Description | Comment |
|---|---|---|
| 0x0A06 | Link Command Failed-Hardware Access Error | Returned when PHY R/W access fails to complete normally while executing the Set Link or Get Link Status command |

### 8.4.25 Set VLAN Filter command (0x0B)

The Set VLAN Filter command is used by the Management Controller to program one or more VLAN IDs that are used for VLAN filtering.

Incoming packets that match both a VLAN ID filter and a MAC address filter are forwarded to the Management Controller. Other packets may be dropped based on the VLAN filtering mode per the Enable VLAN command.

The quantity of each filter type that is supported by the channel can be discovered by means of the Get Capabilities command. Up to 15 filters can be supported per channel. A Network Controller implementation shall support at least one VLAN filter per channel.

To configure a VLAN filter, the Management Controller issues a Set VLAN Filter command with the Filter Selector field indicating which filter is to be configured, the VLAN ID field set to the VLAN TAG values to be used by the filter, and the Enable field set to either enable or disable the selected filter.

The VLAN-related fields are specified per IEEE 802.1q. When VLAN Tagging is used, the packet includes a Tag Protocol Identifier (TPID) field and VLAN Tag fields, as shown in Table 55.

**Table 55 – IEEE 802.1q VLAN Fields**

| Field | Size | Description |
|---|---|---|
| TPI | 2 bytes | Tag Protocol Identifier<br>= 8100h |
| VLAN TAG – user priority | 3 bits | User Priority<br>(typical value = 000b) |
| VLAN TAG – CFI | 1 bit | Canonical Format Indicator = 0b |
| VLAN TAG – VLAN ID | 12 bits | Zeros = no VLAN |

When checking VLAN field values, the Network Controller shall match against the enabled VLAN Tag Filter values that were configured with the S0065t VLAN Filter command. The Network Controller shall also match on the TPI value of 8100h, as specified by IEEE 802.1q. Matching against the User Priority/CFI bits is optional. An implementation may elect to ignore the setting of those fields.

Table 56 illustrates the packet format of the Set VLAN Filter command.

**Table 56 – Set VLAN Filter command packet format**

<table>
<tr><th></th><th colspan="6">Bits</th></tr>
<tr><th>Bytes</th><th>31..24</th><th>23..16</th><th colspan="2">15..08</th><th colspan="2">07..00</th></tr>
<tr><td>00..15</td><td colspan="6">NC-SI Control Packet Header</td></tr>
<tr><td>16..19</td><td colspan="2">Reserved</td><td>User Priority/CFI</td><td colspan="3">VLAN ID</td></tr>
<tr><td>20..23</td><td colspan="2">Reserved</td><td colspan="2">Filter Selector</td><td>Reserved</td><td>E</td></tr>
<tr><td>24..27</td><td colspan="6">Checksum</td></tr>
<tr><td>28..45</td><td colspan="6">Pad</td></tr>
</table>

Table 57 provides possible settings for the Filter Selector field. Table 58 provides possible settings for the Enable (E) field.

**Table 57 – Possible Settings for Filter Selector field (8-bit field)**

| Value | Description |
|---|---|
| 1 | Settings for VLAN filter number 1 |
| 2 | Settings for VLAN filter number 2 |
| .. | |
| N | Settings for VLAN filter number *N* |

**Table 58 – Possible Settings for Enable (E) field (1-bit field)**

| Value | Description |
|---|---|
| 0b | Disable this VLAN filter |
| 1b | Enable this VLAN filter |

### 8.4.26 Set VLAN Filter response (0x8B)

The channel shall, in the absence of a checksum error or identifier mismatch, always accept the Set VLAN Filter command and send a response (see Table 59).

**Table 59 – Set VLAN Filter response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

Table 60 describes the reason code that is specific to the Set VLAN Filter command.

**Table 60 – Set VLAN Filter command-specific reason code**

| Value | Description | Comment |
|---|---|---|
| 0x0B07 | VLAN Tag Is Invalid | Returned when the VLAN ID is invalid (VLAN ID = 0) |

### 8.4.27 Enable VLAN command (0x0C)

The Enable VLAN command may be used by the Management Controller to enable the channel to accept VLAN-tagged packets from the network for NC-SI Pass-through operation (see Table 61).

**Table 61 – Enable VLAN command packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Reserved | | | Mode # |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

Table 62 describes the modes for the Enable VLAN command.

**Table 62 – VLAN Enable modes**

| Mode | # | O/M | Description |
|---|---|---|---|
| Reserved | 0x00 | N/A | Reserved |
| VLAN only | 0x01 | M | Only VLAN-tagged packets that match the enabled VLAN Filter settings (and also match the MAC Address Filtering configuration) are accepted.<br>Non-VLAN-tagged packets are not accepted. |

| | | | |
|---|---|---|---|
| VLAN + non-VLAN | 0x02 | O | VLAN-tagged packets that match the enabled VLAN Filter settings (and also match the MAC Address Filtering configuration) are accepted.<br>Non-VLAN-tagged packets (that also match the MAC Address Filtering configuration) are also accepted. |
| Any VLAN + non-VLAN | 0x03 | O | Any VLAN-tagged packets that also match the MAC Address Filtering configuration are accepted, regardless of the VLAN Filter settings.<br>Non-VLAN-tagged packets (that also match the MAC Address Filtering configuration) are also accepted. |
| Reserved | 0x04 – 0xFF | N/A | Reserved |

### 8.4.28 Enable VLAN response (0x8C)

The channel shall, in the absence of a checksum error or identifier mismatch, always accept the Enable VLAN command and send a response.

Currently no command-specific reason code is identified for this response (see Table 63).

**Table 63 – Enable VLAN response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.4.29 Disable VLAN command (0x0D)

The Disable VLAN command may be used by the Management Controller to disable VLAN filtering. In the disabled state, only non-VLAN-tagged packets (that also match the MAC Address Filtering configuration) are accepted. VLAN-tagged packets are not accepted.

Table 64 illustrates the packet format of the Disable VLAN command.

**Table 64 – Disable VLAN command packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

### 8.4.30 Disable VLAN response (`0x8D`)

The channel shall, in the absence of a checksum error or identifier mismatch, always accept the Disable VLAN command and send a response.

Currently no command-specific reason code is identified for this response (see Table 65).

**Table 65 – Disable VLAN response packet format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.4.31 Set MAC Address command (`0x0E`)

The Set MAC Address command is used by the Management Controller to program the channel's unicast or multicast MAC address filters.

The channel supports one or more "perfect match" MAC address filters that are used to selectively forward inbound frames to the Management Controller. Assuming that a packet passes any VLAN filtering that may be active, it will be forwarded to the Management Controller if its 48-bit destination MAC address exactly matches an active MAC address filter.

MAC address filters may be configured as unicast or multicast addresses, depending on the capability of the channel. The channel may implement three distinct types of filter:

- **Unicast filters** support exact matching on 48-bit unicast MAC addresses (AT = `0x0` only).
- **Multicast filters** support exact matching on 48-bit multicast MAC addresses (AT = `0x1` only).
- **Mixed filters** support matching on both unicast and multicast MAC addresses. (AT = `0x0` or AT = `0x1`)

The number of each type of filter that is supported by the channel can be discovered by means of the Get Capabilities command. The channel shall support at least one unicast address filter or one mixed filter, so that at least one unicast MAC address filter may be configured on the channel. Support for any combination of unicast, multicast, or mixed filters beyond this basic requirement is vendor specific. The total number of all filters shall be less than or equal to 8.

To configure an address filter, the Management Controller issues a Set MAC Address command with the Address Type field indicating the type of address to be programmed (unicast or multicast) and the MAC Address Num field indicating the specific filter to be programmed.

Filters are addressed using a 1-based index ordered over the unicast, multicast, and mixed filters reported by means of the Get Capabilities command. For example, if the interface reports four unicast filters, two multicast filters, and two mixed filters, then MAC Address numbers 1 through 4 refer to the interface's unicast filters, 5 and 6 refer to the multicast filters, and 7 and 8 refer to the mixed filters. Similarly, if the interface reports two unicast filters, no multicast filters, and six mixed filters, then MAC address numbers 1 and 2 refer to the unicast filters, and 3 through 8 refer to the mixed filters.

The filter type of the filter to be programmed (unicast, multicast, or mixed) shall be compatible with the Address Type being programmed. For example, programming a mixed filter to a unicast address is allowed, but programming a multicast filter to a unicast address is an error.

The Enable field determines whether the indicated filter is to be enabled or disabled. When a filter is programmed to be enabled, the filter is loaded with the 48-bit MAC address in the MAC Address field of the command, and the channel enables forwarding of frames that match the configured address. If the specified filter was already enabled, it is updated with the new address provided.

When a filter is programmed to be disabled, the contents of the MAC Address field are ignored. Any previous MAC address programmed in the filter is discarded and the channel no longer uses this filter in its packet-forwarding function.

Only unicast MAC addresses, specified with AT set to `0x0`, should be used in source MAC address checking and for determining the NC-SI channel for Pass-through transmit traffic.

Table 66 illustrates the packet format of the Set MAC Address command.

**Table 66 – Set MAC Address command packet format**

<table>
<tr><th></th><th colspan="6">Bits</th></tr>
<tr><th>Bytes</th><th>31..24</th><th>23..16</th><th>15..08</th><th colspan="3">07..00</th></tr>
<tr><td>00..15</td><td colspan="6">NC-SI Control Packet Header</td></tr>
<tr><td>16..19</td><td>MAC Address byte 5</td><td>MAC Address byte 4</td><td>MAC Address byte 3</td><td colspan="3">MAC Address byte 2</td></tr>
<tr><td>20..23</td><td>MAC Address byte 1</td><td>MAC Address byte 0</td><td>MAC Address Num</td><td>AT</td><td>Rsvd</td><td>E</td></tr>
<tr><td>24..27</td><td colspan="6">Checksum</td></tr>
<tr><td>28..45</td><td colspan="6">Pad</td></tr>
<tr><td colspan="7">NOTE AT = Address Type, E = Enable.</td></tr>
</table>

Table 67 provides possible settings for the MAC Address Number field. Table 68 provides possible settings for the Address Type (AT) field. Table 69 provides possible settings for the Enable (E) field.

**Table 67 – Possible settings for MAC Address Number (8-bit field)**

| Value | Description |
|---|---|
| `0x01` | Configure MAC address filter number 1 |
| `0x02` | Configure MAC address filter number 2 |
| .. | |
| N | Configure MAC address filter number *N* |

**Table 68 – Possible settings for Address Type (3-bit field)**

| Value | Description |
|---|---|
| `0x0` | Unicast MAC address |
| `0x1` | Multicast MAC address |
| `0x2–0x7` | Reserved |

**Table 69 – Possible settings for Enable Field (1-bit field)**

| Value | Description |
|---|---|
| `0b` | Disable this MAC address filter |
| `1b` | Enable this MAC address filter |

### 8.4.32 Set MAC Address response (`0x8E`)

The channel shall, in the absence of a checksum error or identifier mismatch, always accept the Set MAC Address command and send a response (see Table 70).

**Table 70 – Set MAC Address response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

Table 71 describes the reason code that is specific to the Set MAC Address command.

**Table 71 – Set MAC Address command-specific reason code**

| Value | Description | Comment |
|---|---|---|
| `0x0E08` | MAC Address Is Zero | Returned when the Set MAC Address command is received with the MAC address set to 0 |

### 8.4.33 Enable Broadcast Filter command (`0x10`)

The Enable Broadcast Filter command allows the Management Controller to control the forwarding of broadcast frames to the Management Controller. The channel, upon receiving and processing this command, shall filter all received broadcast frames based on the broadcast packet filtering settings specified in the payload. If no broadcast packet types are specified for forwarding, all broadcast packets shall be filtered out.

The Broadcast Packet Filter Settings field is used to specify those protocol-specific broadcast filters that should be activated. The channel indicates which broadcast filters it supports in the Broadcast Filter Capabilities field of the Get Capabilities Response frame defined in clause 8.4.46.

Table 72 illustrates the packet format of the Enable Broadcast Filter command.

2589 **Table 72 – Enable Broadcast Filter command packet format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Broadcast Packet Filter Settings | | | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

2590 Table 73 describes the Broadcast Packet Filter Settings field bit definitions.

2591 **Table 73 – Broadcast Packet Filter Settings field**

| **Bit Position** | **Field Description** | **Value Description** |
|---|---|---|
| 0 | ARP Packets | `1b` = Forward this packet type to the Management Controller.<br>`0b` = Filter out this packet type.<br>For the purposes of this specification, an ARP broadcast packet is defined to be any packet that meets all of the following requirements:<br>• The destination MAC address field is set to the layer 2 broadcast address (`FF:FF:FF:FF:FF:FF`).<br>• The Ethertype field set to `0x0806`.<br>This field is mandatory. |
| 1 | DHCP Client Packets | `1b` = Forward this packet type to the Management Controller.<br>`0b` = Filter out this packet type.<br>For the purposes of this filter, a DHCP client broadcast packet is defined to be any packet that meets all of the following requirements:<br>• The destination MAC address field is set to the layer 2 broadcast address (`FF:FF:FF:FF:FF:FF`).<br>• The Ethertype field is set to `0x0800` (IPv4).<br>• The IP header's Protocol field is set to 17 (UDP).<br>• The UDP destination port number is set to 68.<br>This field is optional. If unsupported, broadcast DHCP client packets will be blocked when broadcast filtering is enabled. The value shall be set to 0 if unsupported. |

| Bit Position | Field Description | Value Description |
|---|---|---|
| 2 | DHCP Server Packets | 1b = Forward this packet type to the Management Controller.<br>0b = Filter out this packet type.<br>For the purposes of this filter, a DHCP server broadcast packet is defined to be any packet that meets all of the following requirements:<br>• The destination MAC address field is set to the layer 2 broadcast address (FF:FF:FF:FF:FF:FF).<br>• The Ethertype field is set to 0x0800 (IPv4).<br>• The IP header's Protocol field is set to 17 (UDP).<br>• The UDP destination port number is set to 67.<br>This field is optional. If unsupported, broadcast DHCP packets will be blocked when broadcast filtering is enabled. The value shall be set to 0b if unsupported. |
| 3 | NetBIOS Packets | 1b = Forward this packet type to the Management Controller.<br>0b = Filter out this packet type.<br>For the purposes of this filter, NetBIOS broadcast packets are defined to be any packet that meets all of the following requirements:<br>• The destination MAC address field is set to the layer 2 broadcast address (FF:FF:FF:FF:FF:FF).<br>• The Ethertype field is set to 0x0800 (IPv4).<br>• The IP header's Protocol field is set to 17 (UDP).<br>• The UDP destination port number is set to 137 for NetBIOS Name Service or 138 for NetBIOS Datagram Service, per the assignment of IANA well-known ports.<br>This field is optional. If unsupported, broadcast NetBIOS packets will be blocked when broadcast filtering is enabled. The value shall be set to 0b if unsupported. |
| 4..31 | Reserved | None |

### 8.4.34 Enable Broadcast Filter response (0x90)

The channel shall, in the absence of a checksum error or identifier mismatch, always accept the Enable Broadcast Filter command and send a response.

Currently no command-specific reason code is identified for this response (see Table 74).

**Table 74 – Enable Broadcast Filter response packet format**

|  | Bits |  |  |  |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header |  |  |  |
| **16..19** | Response Code |  | Reason Code |  |
| **20..23** | Checksum |  |  |  |
| **24..45** | Pad |  |  |  |

### 8.4.35 Disable Broadcast Filter command (`0x11`)

The Disable Broadcast Filter command may be used by the Management Controller to disable the broadcast filter feature and enable the reception of all broadcast frames. Upon processing this command, the channel shall discontinue the filtering of received broadcast frames.

Table 75 illustrates the packet format of the Disable Broadcast Filter command.

**Table 75 – Disable Broadcast Filter command packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

### 8.4.36 Disable Broadcast Filter response (`0x91`)

The channel shall, in the absence of a checksum error or identifier mismatch, always accept the Disable Broadcast Filter command and send a response.

Currently no command-specific reason code is identified for this response (see Table 76).

**Table 76 – Disable Broadcast Filter response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.4.37 Enable Global Multicast Filter command (`0x12`)

The Enable Global Multicast Filter command is used to activate global filtering of multicast frames with optional filtering of specific multicast protocols. Upon receiving and processing this command, the channel shall only deliver multicast frames that match specific multicast MAC addresses enabled for Pass-through using this command or the Set MAC Address command.

The Multicast Packet Filter Settings field is used to specify optional, protocol-specific multicast filters that should be activated. The channel indicates which optional multicast filters it supports in the Multicast Filter Capabilities field of the Get Capabilities Response frame defined in clause 8.4.46. The Management Controller should not set bits in the Multicast Packet Filter Settings field that are not indicated as supported in the Multicast Filter Capabilities field.

Neighbor Solicitation messages are sent to a Solicited Node multicast address that is derived from the target node's IPv6 address. This command may be used to enable forwarding of solicited node multicasts.

The IPv6 neighbor solicitation filter, as defined in this command, may not be supported by the Network Controller. In this case, the Management Controller may configure a multicast or mixed MAC address filter for the specific Solicited Node multicast address using the Set MAC Address command to enable forwarding of Solicited Node multicasts.

This command shall be implemented if the channel implementation supports accepting all multicast addresses. An implementation that does not support accepting all multicast addresses shall not implement these commands. Pass-through packets with multicast addresses can still be accepted depending on multicast address filter support provided by the Set MAC Address command. Multicast filter entries that are set to be enabled in the Set MAC Address command are accepted; all others are rejected. Table 77 illustrates the packet format of the Enable Global Multicast Filter command. Unsupported fields should be treated as reserved fields unless otherwise specified.

**Table 77 – Enable Global Multicast Filter command packet format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Multicast Packet Filter Settings | | | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

Table 78 describes the bit definitions for the Multicast Packet Filter Settings field.

**Table 78 – Bit Definitions for Multicast Packet Filter Settings field**

| **Bit Position** | **Field Description** | **Value Description** |
|---|---|---|
| 0 | IPv6 Neighbor Advertisement | `1b` = Forward this packet type to the Management Controller.<br>`0b` = Filter out this packet type.<br>For the purposes of this specification, an IPv6 Neighbor Advertisement multicast packet is defined to be any packet that meets all of the following requirements:<br>• The destination MAC address field is set to a layer 2 multicast address of the form `33:33:00:00:00:01`. This address corresponds to the all-nodes multicast address (`FF02::1`).<br>• The Ethertype field is set to `0x86DD` (IPv6).<br>• The IPv6 header's Next Header field is set to 58 (ICMPv6).<br>• The ICMPv6 header's Message Type field is set to the following value: 136 – Neighbor Advertisement.<br>This field is optional. |

| Bit Position | Field Description | Value Description |
|---|---|---|
| 1 | IPv6 Router Advertisement | 1b = Forward this packet type to the Management Controller.<br>0b = Filter out this packet type.<br>For the purposes of this specification, an IPv6 Router Advertisement multicast packet is defined to be any packet that meets all of the following requirements:<br>• The destination MAC address field is set to a layer 2 multicast address of the form 33:33:00:00:00:01. This corresponds to the all-nodes multicast address (FF02::1).<br>• The Ethertype field is set to 0x86DD (IPv6).<br>• The IPv6 header's Next Header field is set to 58 (ICMPv6).<br>• The ICMPv6 header's Message Type field is set to 134.<br>This field is optional. |
| 2 | DHCPv6 relay and server multicast | 1b = Forward this packet type to the Management Controller.<br>0b = Filter out this packet type.<br>For the purposes of this filter, a DHCPv6 multicast packet is defined to be any packet that meets all of the following requirements:<br>• The destination MAC address field is set to the layer 2 multicast address 33:33:00:01:00:02 or 33:33:00:01:00:03. These correspond to the IPv6 multicast addresses FF02::1:2 (All_DHCP_Relay_Agents_and_Servers) and FF05::1:3 (All_DHCP_Servers).<br>• The Ethertype field is set to 0x86DD (IPv6).<br>• The IPv6 header's Next Header field is set to 17 (UDP).<br>• The UDP destination port number is set to 547.<br>This field is optional. |
| 3 | DHCPv6 multicasts from server to clients listening on well-known UDP ports | 1b = Forward this packet type to the Management Controller.<br>0b = Filter out this packet type.<br>For the purposes of this filter, a DHCPv6 multicast packet is defined to be any packet that meets all of the following requirements:<br>• The destination MAC address field is set to the layer 2 multicast address 33:33:00:01:00:02. These correspond to the IPv6 multicast addresses FF02::1:2 (All_DHCP_Relay_Agents_and_Servers).<br>• The Ethertype field is set to 0x86DD (IPv6).<br>• The IPv6 header's Next Header field is set to 17 (UDP).<br>• The UDP destination port number is set to 546.<br>This field is optional. |

| Bit Position | Field Description | Value Description |
|---|---|---|
| 4 | IPv6 MLD | 1b = Forward this packet type to the Management Controller.<br>0b = Filter out this packet type.<br>For the purposes of this specification, an IPv6 MLD packet is defined to be any packet that meets all of the following requirements:<br>• The destination MAC address field is set to a layer 2 multicast address of the form 33:33:00:00:00:01. This address corresponds to the All_Nodes (FF02::1) multicast address.<br>• The Ethertype field is set to 0x86DD (IPv6).<br>• The IPv6 header's Next Header field is set to 58 (ICMPv6).<br>• The ICMPv6 header's Message Type field is set to one of the following values: 130 (Multicast Listener Query), 131 (Multicast Listener Report), 132 (Multicast Listener Done)<br>This field is optional. |
| 5 | IPv6 Neighbor Solicitation | 1b = Forward this packet type to the Management Controller.<br>0b = Filter out this packet type.<br>For the purposes of this specification, an IPv6 MLD packet is defined to be any packet that meets all of the following requirements:<br>• The destination MAC address field is set to a layer 2 multicast address of the form 33:33:FF:XX:XX:XX. This address corresponds to the Solicited Note multicast address where the last three bytes of the destination MAC address are ignored for this filter.<br>• The Ethertype field is set to 0x86DD (IPv6).<br>• The IPv6 header's Next Header field is set to 58 (ICMPv6).<br>• The ICMPv6 header's Message Type field is set to one of the following values: 135<br>This field is optional.<br>Implementation Note: Enabling of this filter results in receiving all IPv6 neighbor solicitation traffic on this channel. If IPv6 neighbor solicitation traffic for a specific multicast address is of interest, then it is recommended that the MC uses a multicast address filter (configured for the multicast address using the Set MAC Address command) instead of this filter. |

| Bit Position | Field Description | Value Description |
|---|---|---|
| 6 | LLDP | 1b = Forward this packet type to the Management Controller.<br>0b = Filter out this packet type.<br>For the purposes of this specification, a LLDP packet is defined to be any packet that meets all of the following requirements:<br>The destination MAC address field is set to a layer 2 multicast address of the form 01:80:C2:00:00:00, or 01:80:C2:00:00:03, or 01:80:C2:00:00:0E.<br>The Ethertype field is set to 0x88CC.<br>This field is optional.<br>Implementation Note: Enabling of this filter results in receiving a copy of all LLDP traffic on this channel. If LLDP traffic for a specific LLDP multicast address is of interest, then it is recommended that the MC uses a multicast address filter (configured for the multicast address using the Set MAC Address command) instead of this filter.<br>The intent of this filter is to allow the MC to snoop the received LLDP frame by the port, not to achieve ownership of any contained protocols. |
| 7 | mDNSv4 | 1b = Forward this packet type to the Management Controller.<br>0b = Filter out this packet type.<br>For the purposes of this specification, a mDNS/IPv4 packet is defined to be any packet that meets all the following requirements:<br>The destination MAC address field is set to a layer 2 multicast address of the form 01:00:5E:00:00:FB.<br>The Ethertype field is set to 0x0800.<br>The IPv4 address is 224.0.0.251.<br>The IPv4 header's Protocol field is set to 17 (UDP).<br>The UDP destination port number is set to 5353.<br>This field is optional. |
| 8 | mDNSv6 | 1b = Forward this packet type to the Management Controller.<br>0b = Filter out this packet type.<br>For the purposes of this specification, a mDNS/IPv6 packet is defined to be any packet that meets all the following requirements:<br>The destination MAC address field is set to a layer 2 multicast address of the form 33:33:00:00:00:FB. This corresponds to the All Nodes IPv6 multicast address, FF02::FB.<br>The Ethertype field is set to 0x086DD.<br>The IPv6 header's Next Header field is set to 17 (UDP).<br>The UDP destination port number is set to 5353.<br>This field is optional. |
| 31..9 | Reserved | None |

### 8.4.38 Enable Global Multicast Filter response (`0x92`)

The channel shall, in the absence of a checksum error or identifier mismatch, always accept the Enable Global Multicast Filter command and send a response.

Currently no command-specific reason code is identified for this response (see Table 79).

**Table 79 – Enable Global Multicast Filter response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.4.39 Disable Global Multicast Filter command (`0x13`)

The Disable Global Multicast Filter command is used to disable global filtering of multicast frames. Upon receiving and processing this command, and regardless of the current state of multicast filtering, the channel shall forward all multicast frames to the Management Controller.

This command shall be implemented on the condition that the channel implementation supports accepting all multicast addresses. An implementation that does not support accepting all multicast addresses shall not implement these commands. Pass-through packets with multicast addresses can still be accepted depending on multicast address filter support provided by the Set MAC Address command. Packets with destination addresses matching multicast filter entries that are set to enabled in the Set MAC Address command are accepted; all others are rejected.

Table 80 illustrates the packet format of the Disable Global Multicast Filter command.

**Table 80 – Disable Global Multicast Filter command packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

### 8.4.40 Disable Global Multicast Filter response (`0x93`)

In the absence of any errors, the channel shall process and respond to the Disable Global Multicast Filter command by sending the response packet shown in Table 81.

Currently no command-specific reason code is identified for this response.

**Table 81 – Disable Global Multicast Filter response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.4.41 Set NC-SI Flow Control command (`0x14`)

The Set NC-SI Flow Control command allows the Management Controller to configure IEEE 802.3 pause packet flow control on the NC-SI RBT.

The Set NC-SI Flow Control command is addressed to the package, rather than to a particular channel (that is, the command is sent with a Channel ID where the Package ID subfield matches the ID of the intended package and the Internal Channel ID subfield is set to `0x1F`).

The setting of IEEE 802.3 Pause packet flow control on RBT is independent from any arbitration scheme, if any is used.

Table 82 illustrates the packet format of the Set NC-SI Flow Control command.

**Table 82 – Set NC-SI Flow Control command packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Reserved | | | Flow Control Enable |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

Table 83 describes the values for the Flow Control Enable field.

**Table 83 – Values for the Flow Control Enable field (8-bit field)**

| Value | Description |
|---|---|
| `0x0` | Disables NC-SI flow control |
| `0x1` | Enables Network Controller to Management Controller flow control frames (Network Controller generates flow control frames)<br>This field is optional. |
| `0x2` | Enables Management Controller to Network Controller flow control frames (Network Controller accepts flow control frames)<br>This field is optional. |
| `0x3` | Enables bi-directional flow control frames<br>This field is optional. |

| Value | Description |
|---|---|
| 0x4..0xFF | Reserved |

### 8.4.42 Set NC-SI Flow Control response (0x94)

The package shall, in the absence of a checksum error or identifier mismatch, always accept the Set NC-SI Flow Control command and send a response (see Table 84).

**Table 84 – Set NC-SI Flow Control response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

Table 85 describes the reason code that is specific to the Set NC-SI Flow Control command.

**Table 85 – Set NC-SI Flow Control command-specific reason code**

| Value | Description | Comment |
|---|---|---|
| 0x1409 | Independent transmit and receive enable/disable control is not supported | Returned when the implementation requires that both transmit and receive flow control be enabled and disabled simultaneously |

### 8.4.43 Get Version ID command (0x15)

The Get Version ID command may be used by the Management Controller to request the channel to provide the controller and firmware type and version strings listed in the response payload description.

Table 86 illustrates the packet format of the Get Version ID command.

**Table 86 – Get Version ID command packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

### 8.4.44 Get Version ID Response (0x95)

The channel shall, in the absence of an error, always accept the Get Version ID command and send the response packet shown in Table 87. Currently no command-specific reason code is identified for this response.

NOTE: When multiple Physical Functions are enabled on the channel, the PCI ID that is returned shall be that of the lowest numbered Function on the channel.

**Table 87 – Get Version ID response packet format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | NC-SI Version | | | |
| | Major | Minor | Update | Alpha1 |
| **24..27** | Reserved | reserved | reserved | Alpha2 |
| **28..31** | Firmware Name String (11-08) | | | |
| **32..35** | Firmware Name String (07-04) | | | |
| **36..39** | Firmware Name String (03-00) | | | |
| **40..43** | Firmware Version | | | |
| | MS-byte (3) | Byte (2) | Byte (1) | LS-byte (0) |
| **44..47** | PCI DID | | PCI VID | |
| **48..51** | PCI SSID | | PCI SVID | |
| **52..55** | Manufacturer ID (IANA) | | | |
| **56..59** | Checksum | | | |

### 8.4.44.1 NC-SI Version encoding

The NC-SI Version field holds the version number of the NC-SI specification with which the controller is compatible. The version field shall be encoded as follows:

- The ‘major’, ‘minor’, and ‘update’ bytes are BCD-encoded, and each byte holds two BCD digits.
- The ‘alpha’ byte holds an optional alphanumeric character extension that is encoded using the ISO/IEC 8859-1 Character Set.
- The semantics of these fields follow the semantics specified in DSP4014.
- The value `0x00` in the Alpha1 or Alpha2 fields means that the corresponding alpha field is not used. The Alpha1 field shall be used first.
- The value `0xF` in the most-significant nibble of a BCD-encoded value indicates that the most-significant nibble should be ignored and the overall field treated as a single digit value.
- A value of `0xFF` in the update field indicates that the entire field is not present. `0xFF` is not allowed as a value for the major or minor fields.

EXAMPLE: Version 3.7.10a → `0xF3F7106100`
Version 10.01.7 → `0x1001F70000`
Version 3.1 → `0xF3F1FF0000`
Version 1.0a → `0xF1F0FF4100`
Version 1.0ab → `0xF1F0FF4142 (Alpha1 = 0x41, Alpha2 = 0x42)`

NC-SI implementations that follow this particular specification shall return the following version information in the response.

The Major field shall be set to 0xF1 to indicate compatibility with Version 1.0 of the NC-SI specification.

The Minor field shall be set to 0xF2 to indicate compatibility with Version 1.2 of the NC-SI specification.

The Update field shall be set to 0xF0 to indicate compatibility with Version 1.2.0 of the NC-SI specification.

The Alpha1 field shall be set to 0x00.

The Alpha2 field shall be set to 0x00.

The reported NC-SI version using the encoding shall be: 0xF1F2F00000 (1.2.0).

#### 8.4.44.2 Firmware Name encoding

The Firmware Name String shall be encoded using the ISO/IEC 8859-1 Character Set. Strings are left-justified where the leftmost character of the string occupies the most-significant byte position of the Firmware Name String field, and characters are populated starting from that byte position. The string is null terminated if the string is smaller than the field size. That is, the delimiter value, `0x00`, follows the last character of the string if the string occupies fewer bytes than the size of the field allows. A delimiter is not required if the string occupies the full size of the field. Bytes following the delimiter (if any) should be ignored and can be any value.

#### 8.4.44.3 Firmware Version encoding

To facilitate a common way of representing and displaying firmware version numbers across different vendors, each byte is hexadecimal encoded where each byte in the field holds two hexadecimal digits. The Firmware Version field shall be encoded as follows. The bytes are collected into a single 32-bit field where each byte represents a different 'point number' of the overall version. The selection of values that represent a particular version of firmware is specific to the Network Controller vendor.

Software displaying these numbers should not suppress leading zeros, which should help avoid user confusion in interpreting the numbers. For example, consider the two values `0x05` and `0x31`. Numerically, the byte `0x31` is greater that `0x05`, but if leading zeros were incorrectly suppressed, the two displayed values would be ".5" and ".31", respectively, and a user would generally interpret 0.5 as representing a greater value than 0.31 instead of 0.05 being smaller than 0.31. Similarly, if leading zeros were incorrectly suppressed, the value `0x01` and `0x10` would be displayed as 0.1 and 0.10, which could potentially be misinterpreted as representing the same version instead of 0.01 and 0.10 versions.

EXAMPLE: `0x00030217` → Version 00.03.02.17
`0x010100A0` → Version 01.01.00.A0

#### 8.4.44.4 PCI ID fields

These fields (PCI DID, PCI VID, PCI SSID, PCI SVID) hold the PCI ID information for the Network Controller when the Network Controller incorporates a PCI or PCI Express™ interface that provides a host network interface connection that is shared with the NC-SI connection to the network.

If this field is not used, the values shall all be set to zeros (`0000h`). Otherwise, the fields shall hold the PCI ID information for the host interface as defined by the version of the PCI/PCI Express™ specification to which the device's interface was designed.

If multiple partitions are enabled on the channel, the values should represent the PCI ID of the lowest Function number assigned to the channel by the Set PF Assignment command `(0x28)`.

#### 8.4.44.5 Manufacturer ID (IANA) field

The Manufacturer ID holds the IANA Enterprise Number for the manufacturer of the Network Controller as a 32-bit binary number. If the field is unused, the value shall be set to `0xFFFFFFFF`.

### 8.4.45 Get Capabilities command (`0x16`)

The Get Capabilities command is used to discover additional optional functions supported by the channel, such as the number of unicast/multicast addresses supported, the amount of buffering in bytes available for packets bound for the Management Controller, and so on.

Table 88 illustrates the packet format for the Get Capabilities command.

**Table 88 – Get Capabilities command packet format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

### 8.4.46 Get Capabilities response (`0x96`)

In the absence of any errors, the channel shall process and respond to the Get Capabilities Command and send the response packet shown in Table 89. Currently no command-specific reason code is identified for this response.

**Table 89 – Get Capabilities response packet format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Capabilities Flags | | | |
| **24..27** | Broadcast Packet Filter Capabilities | | | |
| **28..31** | Multicast Packet Filter Capabilities | | | |
| **32..35** | Buffering Capability | | | |
| **36..39** | AEN Control Support | | | |
| **40..43** | VLAN Filter Count | Mixed Filter Count | Multicast Filter Count | Unicast Filter Count |
| **44..47** | Reserved | | VLAN Mode Support | Channel Count |
| **48..51** | Checksum | | | |

#### 8.4.46.1 Capabilities Flags field

The Capabilities Flags field indicates which optional features of this specification the channel supports, as described in Table 90.

**Table 90 – Capabilities Flags bit definitions**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | Hardware Arbitration Capability | 0b = Hardware arbitration capability is not supported by the package.<br>1b = Hardware arbitration capability is supported by the package. |
| 1 | Host NC Driver Status | 0b = Host NC Driver Indication status is not supported.<br>1b = Host NC Driver Indication status is supported.<br>See Table 52 for the definition of Host NC Driver Indication Status. |
| 2 | Network Controller to Management Controller Flow Control Support | 0b = Network Controller to Management Controller flow control is not supported.<br>1b = Network Controller to Management Controller flow control is supported. |
| 3 | Management Controller to Network Controller Flow Control Support | 0b = Management Controller to Network Controller flow control is not supported.<br>1b = Management Controller to Network Controller flow control is supported. |
| 4 | All multicast addresses support | 0b = The channel cannot accept all multicast addresses. The channel does not support enable/disable global multicast commands.<br>1b = The channel can accept all multicast addresses. The channel supports enable/disable global multicast commands. |
| 6..5 | Hardware Arbitration<br>Implementation Status | 00b = Unknown<br>01b = Hardware arbitration capability is not implemented for the package on the given system.<br>10b = Hardware arbitration capability is implemented for the package on the given system.<br>11b = Reserved. |
| 7 | Thermal shutdown Implementation Status | 0b = The thermal self-shutdown capability is not supported by the channel (package).<br>1b = The thermal self-shutdown capability is supported by the channel (package). |
| 8 | Delayed Response Support | 0b = Delayed response operation and signaling is not supported by the channel (package).<br>1b = Delayed response operation and signaling is supported by the channel (package). |
| 9..31 | Reserved | Reserved |

### 8.4.46.2 Broadcast Packet Filter Capabilities field

The Broadcast Packet Filter Capabilities field defines the optional broadcast packet filtering capabilities that the channel supports. The bit definitions for this field correspond directly with the bit definitions for the Broadcast Packet Filter Settings field defined for the Enable Broadcast Filter command in Table 73. A bit set to 1 indicates that the channel supports the filter associated with that bit position; otherwise, the channel does not support that filter.

#### 8.4.46.3 Multicast Packet Filter Capabilities field

The Multicast Packet Filter Capabilities field defines the optional multicast packet filtering capabilities that the channel supports. The bit definitions for this field correspond directly with the bit definitions for the Multicast Packet Filter Settings field defined for the Enable Global Multicast Filter command in Table 78. A bit set to 1 indicates that the channel supports the filter associated with that bit position; otherwise, the channel does not support that filter.

#### 8.4.46.4 Buffering Capability field

The Buffering Capability field defines the amount of buffering in bytes that the channel provides for inbound packets destined for the Management Controller. The Management Controller may make use of this value in software-based Device Selection implementations to determine the relative time for which a specific channel may be disabled before it is likely to start dropping packets. A value of 0 indicates that the amount of buffering is unspecified.

#### 8.4.46.5 AEN Control Support field

The AEN Control Support field indicates various standard AENs supported by the implementation. The format of the field is shown in Table 42.

#### 8.4.46.6 VLAN Filter Count field

The VLAN Filter Count field indicates the number of VLAN filters, up to 15, that the channel supports, as defined by the Set VLAN Filter command.

#### 8.4.46.7 Mixed, Multicast, and Unicast Filter Count fields

The Mixed Filter Count field indicates the number of mixed address filters that the channel supports. A mixed address filter can be used to filter on specific unicast or multicast MAC addresses.

The Multicast Filter Count field indicates the number of multicast MAC address filters that the channel supports.

The Unicast Filter Count field indicates the number of unicast MAC address filters that the channel supports.

The channel is required to support at least one unicast or mixed filter, such that at least one unicast MAC address can be configured on the interface. The total number of unicast, multicast, and mixed filters shall not exceed 8.

#### 8.4.46.8 VLAN Mode Support field

The VLAN Mode Support field indicates various modes supported by the implementation. The format of field is defined in Table 91.

**Table 91 – VLAN Mode Support bit definitions**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | VLAN only | 1 = VLAN shall be supported in the implementation. |
| 1 | VLAN + non-VLAN | 0 = Filtering ‘VLAN + non-VLAN’ traffic is not supported in the implementation.<br>1 = Filtering ‘VLAN + non-VLAN’ traffic is supported in the implementation. |

| Bit Position | Field Description | Value Description |
|---|---|---|
| 2 | Any VLAN + non-VLAN | 0 = Filtering ‘Any VLAN + non-VLAN’ traffic is not supported in the implementation.<br>1 = Filtering ‘Any VLAN + non-VLAN’ traffic is supported in the implementation. |
| 3..7 | Reserved | 0 |

#### 8.4.46.9 Channel Count field

The Channel Count field indicates the number of channels supported by the Network Controller.

### 8.4.47 Get Parameters command (`0x17`)

The Get Parameters command can be used by the Management Controller to request that the channel send the Management Controller a copy of all of the currently stored parameter settings that have been put into effect by the Management Controller, plus “other” Host/Channel parameter values that may be added to the Get Parameters Response Payload.

Table 92 illustrates the packet format for the Get Parameters command.

**Table 92 – Get Parameters command packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

### 8.4.48 Get Parameters response (`0x97`)

The channel shall, in the absence of a checksum error or identifier mismatch, always accept the Get Parameters command and send a response. As shown in Table 93, each parameter shall return the value that was set by the Management Controller. If the parameter is not supported, 0 is returned. Currently no command-specific reason code is identified for this response.

The payload length of this response packet will vary according to how many MAC address filters or VLAN filters the channel supports. All supported MAC addresses are returned at the end of the packet, without any intervening padding between MAC addresses.

MAC addresses are returned in the following order: unicast filtered addresses first, followed by multicast filtered addresses, followed by mixed filtered addresses, with the number of each corresponding to those reported through the Get Capabilities command. For example, if the interface reports four unicast filters, two multicast filters, and two mixed filters, then MAC addresses 1 through 4 are those currently configured through the interface’s unicast filters, MAC addresses 5 and 6 are those configured through the multicast filters, and 7 and 8 are those configured through the mixed filters. Similarly, if the interface reports two unicast filters, no multicast filters, and six mixed filters, then MAC addresses 1 and 2 are those currently configured through the unicast filters, and 3 through 8 are those configured through the mixed filters.

**Table 93 – Get Parameters response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | MAC Address Count | Reserved | | MAC Address Flags |
| **24..27** | VLAN Tag Count | Reserved | VLAN Tag Flags | |
| **28..31** | Link Settings | | | |
| **32..35** | Broadcast Packet Filter Settings | | | |
| **36..39** | Configuration Flags | | | |
| **40..43** | VLAN Mode | Flow Control Enable | Reserved | |
| **44..47** | AEN Control | | | |
| **48..51** | MAC Address 1 byte 5 | MAC Address 1 byte 4 | MAC Address 1 byte 3 | MAC Address 1 byte 2 |
| **52..55**[a] | MAC Address 1 byte 1 | MAC Address 1 byte 0 | MAC Address 2 byte 5 | MAC Address 2 byte 4 |
| **56..59** | MAC Address 2 byte 3 | MAC Address 2 byte 2 | MAC Address 2 byte 1 | MAC Address 2 byte 0 |
| **variable** | … | | | |
| | VLAN Tag 1 | | VLAN Tag 2 | |
| | … | | | |
| | … | | Pad (if needed) | |
| | Checksum | | | |

[a] Variable fields can start at this byte offset.

Table 94 lists the parameters for which values are returned in this response packet.

**Table 94 – Get Parameters data definition**

| Parameter Field Name | Description |
|---|---|
| MAC Address Count | The number of MAC addresses supported by the channel |
| MAC Address Flags | The enable/disable state for each supported MAC address<br>See Table 95. |
| VLAN Tag Count | The number of VLAN Tags supported by the channel |
| VLAN Tag Flags | The enable/disable state for each supported VLAN Tag<br>See Table 96. |
| Link Settings | The 32-bit Link Settings value as defined in the Set Link command. See Table 45. |
| Broadcast Packet Filter Settings | The current 32-bit Broadcast Packet Filter Settings value |
| Configuration Flags | See Table 97. |

| Parameter Field Name | Description |
|---|---|
| VLAN Mode | See Table 62. |
| Flow Control Enable | See Table 83. |
| AEN Control | See Table 42. |
| MAC Address 1..8 | The current contents of up to eight 6-byte MAC address filter values. |
| VLAN Tag 1..15 | The current contents of up to 15 16-bit VLAN Tag filter values |
| … | |

2832 The format of the MAC Address Flags field is defined in Table 95.

2833 **Table 95 – MAC Address Flags bit definitions**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | MAC address 1 status | `0b` = Default or unsupported or disabled<br>`1b` = Enabled |
| 1 | MAC address 2 status, or Reserved | `0b` = Default or unsupported or disabled<br>`1b` = Enabled |
| 2 | MAC address 3 status, or Reserved | `0b` = Default or unsupported or disabled<br>`1b` = Enabled |
| … | … | … |
| 7 | MAC address 8 status, or Reserved | `0b` = Default or unsupported or disabled<br>`1b` = Enabled |

2834 The format of the VLAN Tag Flags field is defined in Table 96.

2835 **Table 96 – VLAN Tag Flags bit definitions**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | VLAN Tag 1 status | `0b` = Default or unsupported or disabled<br>`1b` = Enabled |
| 1 | VLAN Tag 2 status, or Reserved | `0b` = Default or unsupported or disabled<br>`1b` = Enabled |
| 2 | VLAN Tag 3 status, or Reserved | `0b` = Default or unsupported or disabled<br>`1b` = Enabled |
| … | … | … |
| 14 | VLAN Tag 15 status, or Reserved | `0b` = Default or unsupported or disabled<br>`1b` = Enabled |

2836 The format of the Configuration Flags field is defined in Table 97.

**Table 97 – Configuration Flags bit definitions**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | Broadcast Packet Filter status | 0b = Disabled<br>1b = Enabled |
| 1 | Channel Enabled | 0b = Disabled<br>1b = Enabled |
| 2 | Channel Network TX Enabled | 0b = Disabled<br>1b = Enabled |
| 3 | Global Multicast Packet Filter Status | 0b = Disabled<br>1b = Enabled |
| 4..31 | Reserved | Reserved |

### 8.4.49 Get Controller Packet Statistics command (0x18)

The Get Controller Packet Statistics command may be used by the Management Controller to request a copy of the aggregated Ethernet packet statistics that the channel maintains for its external interface to the LAN network. The statistics are an aggregation of statistics for both the host side traffic and the NC-SI Pass-through traffic.

**Table 98 – Get Controller Packet Statistics command packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

### 8.4.50 Get Controller Packet Statistics response (0x98)

The channel shall, in the absence of a checksum error or identifier mismatch, always accept the Get Controller Packet Statistics command and send the response packet shown in Table 99.

The Get Controller Packet Statistics Response frame contains a set of Ethernet statistics counters that monitor the LAN traffic in the Network Controller. Implementation of the counters listed in Table 100 is optional. The Network Controller shall return any unsupported counter with a value of 0xFFFFFFFF for 32-bit counters and 0xFFFFFFFFFFFFFFFF for 64-bit counters.

**Table 99 – Get Controller Packet Statistics response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Counters Cleared from Last Read (MS Bits) | | | |
| **24..27** | Counters Cleared from Last Read (LS Bits) | | | |
| **28..35** | Total Bytes Received | | | |
| **36..43** | Total Bytes Transmitted | | | |
| **44..51** | Total Unicast Packets Received | | | |
| **52..59** | Total Multicast Packets Received | | | |
| **60..67** | Total Broadcast Packets Received | | | |
| **68..75** | Total Unicast Packets Transmitted | | | |
| **76..83** | Total Multicast Packets Transmitted | | | |
| **84..91** | Total Broadcast Packets Transmitted | | | |
| **92..95** | FCS Receive Errors | | | |
| **96..99** | Alignment Errors | | | |
| **100..103** | False Carrier Detections | | | |
| **104..107** | Runt Packets Received | | | |
| **108..111** | Jabber Packets Received | | | |
| **112..115** | Pause XON Frames Received | | | |
| **116..119** | Pause XOFF Frames Received | | | |
| **120..123** | Pause XON Frames Transmitted | | | |
| **124..127** | Pause XOFF Frames Transmitted | | | |
| **128..131** | Single Collision Transmit Frames | | | |
| **132..135** | Multiple Collision Transmit Frames | | | |
| **136..139** | Late Collision Frames | | | |
| **140..143** | Excessive Collision Frames | | | |
| **144..147** | Control Frames Received<br>For version 1.2, this counter may include Priority flow control packets | | | |
| **148..151** | 64-Byte Frames Received | | | |
| **152..155** | 65–127 Byte Frames Received | | | |
| **156..159** | 128–255 Byte Frames Received | | | |
| **160..163** | 256–511 Byte Frames Received | | | |
| **164..167** | 512–1023 Byte Frames Received | | | |
| **168..171** | 1024–1522 Byte Frames Received | | | |
| **172..175** | 1523–9022 Byte Frames Received | | | |

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **176..179** | 64-Byte Frames Transmitted | | | |
| **180..183** | 65–127 Byte Frames Transmitted | | | |
| **184..187** | 128–255 Byte Frames Transmitted | | | |
| **188..191** | 256–511 Byte Frames Transmitted | | | |
| **192..195** | 512–1023 Byte Frames Transmitted | | | |
| **196..199** | 1024–1522 Byte Frames Transmitted | | | |
| **200..203** | 1523–9022 Byte Frames Transmitted | | | |
| **204..211** | Valid Bytes Received | | | |
| **212..215** | Error Runt Packets Received | | | |
| **216..219** | Error Jabber Packets Received | | | |
| **220..223** | Checksum | | | |

**Table 100 – Get Controller Packet Statistics counters**

| Counter Number | Name | Meaning |
|---|---|---|
| 0 | Total Bytes Received | Counts the number of bytes received |
| 1 | Total Bytes Transmitted | Counts the number of bytes transmitted |
| 2 | Total Unicast Packets Received | Counts the number of good (FCS valid) packets received that passed L2 filtering by a specific MAC address |
| 3 | Total Multicast Packets Received | Counts the number of good (FCS valid) multicast packets received |
| 4 | Total Broadcast Packets Received | Counts the number of good (FCS valid) broadcast packets received |
| 5 | Total Unicast Packets Transmitted | Counts the number of good (FCS valid) packets transmitted that passed L2 filtering by a specific MAC address |
| 6 | Total Multicast Packets Transmitted | Counts the number of good (FCS valid) multicast packets transmitted |
| 7 | Total Broadcast Packets Transmitted | Counts the number of good (FCS valid) broadcast packets transmitted |
| 8 | FCS Receive Errors | Counts the number of receive packets with FCS errors |
| 9 | Alignment Errors | Counts the number of receive packets with alignment errors |
| 10 | False Carrier Detections | Counts the false carrier errors reported by the PHY |
| 11 | Runt Packets Received | Counts the number of received frames that passed address filtering, were less than minimum size (64 bytes from <Destination Address> through <FCS>, inclusively), and had a valid FCS |

| Counter Number | Name | Meaning |
|---|---|---|
| 12 | Jabber Packets Received | Counts the number of received frames that passed address filtering, were greater than the maximum size, and had a valid FCS |
| 13 | Pause XON Frames Received | Counts the number of XON packets received from the network |
| 14 | Pause XOFF Frames Received | Counts the number of XOFF packets received from the network |
| 15 | Pause XOFF Frames Transmitted | Counts the number of XON packets transmitted to the network |
| 16 | Pause XOFF Frames Transmitted | Counts the number of XOFF packets transmitted to the network |
| 17 | Single Collision Transmit Frames | Counts the number of times that a successfully transmitted packet encountered a single collision |
| 18 | Multiple Collision Transmit Frames | Counts the number of times that a transmitted packet encountered more than one collision but fewer than 16 |
| 19 | Late Collision Frames | Counts the number of collisions that occurred after one slot time (defined by IEEE 802.3) |
| 20 | Excessive Collision Frames | Counts the number of times that 16 or more collisions occurred on a single transmit packet |
| 21 | Control Frames Received | Counts the number of MAC control frames received that are *not* XON or XOFF flow control frames |
| 22 | 64 Byte Frames Received | Counts the number of good packets received that are exactly 64 bytes (from <Destination Address> through <FCS>, inclusively) in length |
| 23 | 65–127 Byte Frames Received | Counts the number of good packets received that are 65–127 bytes (from <Destination Address> through <FCS>, inclusively) in length |
| 24 | 128–255 Byte Frames Received | Counts the number of good packets received that are 128–255 bytes (from <Destination Address> through <FCS>, inclusively) in length |
| 25 | 256–511 Byte Frames Received | Counts the number of good packets received that are 256–511 bytes (from <Destination Address> through <FCS>, inclusively) in length |
| 26 | 512–1023 Byte Frames Received | Counts the number of good packets received that are 512–1023 bytes (from <Destination Address> through <FCS>, inclusively) in length |
| 27 | 1024–1522 Byte Frames Received | Counts the number of good packets received that are 1024–1522 bytes (from <Destination Address> through <FCS>, inclusively) in length |
| 28 | 1523–9022 Byte Frames Received | Counts the number of received frames that passed address filtering and were greater than 1523 bytes in length |
| 29 | 64 Byte Frames Transmitted | Counts the number of good packets transmitted that are exactly 64 bytes (from <Destination Address> through <FCS>, inclusively) in length |

| Counter Number | Name | Meaning |
|---|---|---|
| 30 | 65–127 Byte Frames Transmitted | Counts the number of good packets transmitted that are 65–127 bytes (from <Destination Address> through <FCS>, inclusively) in length |
| 31 | 128–255 Byte Frames Transmitted | Counts the number of good packets transmitted that are 128–255 bytes (from <Destination Address> through <FCS>, inclusively) in length |
| 32 | 256–511 Byte Frames Transmitted | Counts the number of good packets transmitted that are 256–511 bytes (from <Destination Address> through <FCS>, inclusively) in length |
| 33 | 512–1023 Byte Frames Transmitted | Counts the number of good packets transmitted that are 512–1023 bytes (from <Destination Address> through <FCS>, inclusively) in length |
| 34 | 1024–1522 Byte Frames Transmitted | Counts the number of good packets transmitted that are 1024–1522 bytes (from <Destination Address> through <FCS>, inclusively) in length |
| 35 | 1523–9022 Byte Frames Transmitted | Counts the number of transmitted frames that passed address filtering and were greater than 1523 in length |
| 36 | Valid Bytes Received | Counts the bytes received in all packets that did not manifest any type of error |
| 37 | Error Runt Packets Received | Counts the number of invalid frames that were less than the minimum size (64 bytes from <Destination Address> through <FCS>, inclusively) |
| 38 | Error Jabber Packets Received | Counts Jabber packets, which are defined as packets that exceed the programmed MTU size *and* have a bad FCS value |

The Network Controller shall also indicate in the Counters Cleared from Last Read fields whether the corresponding field has been cleared by means other than NC-SI (possibly by the host) since it was last read by means of the NC-SI. Counting shall resume from 0 after a counter has been cleared. The Counters Cleared from Last Read field’s format is shown in Table 101.

Currently no command-specific reason code is identified for this response.

**Table 101 – Counters Cleared from Last Read Fields format**

| Field | Bits | Mapped to Counter Numbers |
|---|---|---|
| MS Bits | 0..6 | 32..38 |
| | 7..31 | Reserved |
| LS Bits | 0..31 | 0..31 |

Implementation Note: The Get Controller Packet Statistics response contains the following counters related to flow control: Pause XON Frames Received, Pause XOFF Frames Received, Pause XON Frames Transmitted, and Pause XOFF Frames Transmitted. An implementation can optionally include Priority-Based Flow Control (PFC) packets in these counters.

### 8.4.51 Get NC-SI Statistics command (`0x19`)

In addition to the packet statistics accumulated on the LAN network interface, the channel separately accumulates a variety of NC-SI specific packet statistics for the channel. The Get NC-SI Statistics command may be used by the Management Controller to request that the channel send a copy of all current NC-SI packet statistic values for the channel. The implementation may or may not include statistics for commands that are directed to the package.

Table 102 illustrates the packet format of the Get NC-SI Statistics command.

**Table 102 – Get NC-SI Statistics command packet format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

## 8.4.52 Get NC-SI Statistics response (`0x99`)

In the absence of any error, the channel shall process and respond to the Get NC-SI Statistics command by sending the response packet and payload shown in Table 103.

**Table 103 – Get NC-SI Statistics response packet format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | NC-SI Commands Received | | | |
| **24..27** | NC-SI Control Packets Dropped | | | |
| **28..31** | NC-SI Command Type Errors | | | |
| **32..35** | NC-SI Command Checksum Errors | | | |
| **36..39** | NC-SI Receive Packets | | | |
| **40..43** | NC-SI Transmit Packets | | | |
| **44..47** | AENs Sent | | | |
| **48..51** | Checksum | | | |

The Get NC-SI Statistics Response frame contains a set of statistics counters that monitor the NC-SI traffic in the Network Controller. Counters that are supported shall be reset to `0x0` when entering the Initial State and after being read. Implementation of the counters shown in Table 104 is optional. The Network Controller shall return any unsupported counter with a value of `0xFFFFFFFF`. Counters may wraparound or stop if they reach `0xFFFFFFFE`. It is vendor-specific how NC-SI commands that are sent to the package ID are included in the NC-SI statistics.

Currently no command-specific reason code is identified for this response.

**Table 104 – Get NC-SI Statistics counters**

| Counter Number | Name | Meaning |
|---|---|---|
| 1 | NC-SI Commands Received | For packets that are not dropped, this field returns the number of NC-SI Control Packets received and identified as NC-SI commands. |
| 2 | NC-SI Control Packets Dropped | Counts the number of NC-SI Control Packets that were received and dropped (Packets with correct FCS and Ethertype, but are dropped for one of the other reasons listed in clause 6.8.2.1). NC-SI Control Packets that were dropped because the channel ID was not valid may not be included in this statistics counter. |
| 3 | NC-SI Unsupported Commands Received | Counts the number of NC-SI command packets that were received but are not supported. (Network controller responded to the command with a Command Unsupported response code). |
| 4 | NC-SI Command Checksum Errors | Counts the number of NC-SI Control Packets that were received but dropped because of an invalid checksum (if checksum is provided and checksum validation is supported by the channel) |
| 5 | NC-SI Receive Packets | Counts the total number of NC-SI Control Packets received. This count is the sum of NC-SI Commands Received and NC-SI Control Packets Dropped. |
| 6 | NC-SI Transmit Packets | Counts the total number of NC-SI Control Packets transmitted to the Management Controller. This count is the sum of NC-SI responses sent and AENs sent. |
| 7 | AENs Sent | Counts the total number of AEN packets transmitted to the Management Controller |

### 8.4.53 Get NC-SI Pass-through Statistics command (`0x1A`)

The Get NC-SI Pass-through Statistics command may be used by the Management Controller to request that the channel send a copy of all current NC-SI Pass-through packet statistic values.

Table 105 illustrates the packet format of the Get NC-SI Pass-through Statistics command.

**Table 105 – Get NC-SI Pass-through Statistics command packet format**

| | Bits | | | |
|---|---|---|---|---|
| Bytes | 31..24 | 23..16 | 15..08 | 07..00 |
| 00..15 | NC-SI Control Packet Header | | | |
| 16..19 | Checksum | | | |
| 20..45 | Pad | | | |

### 8.4.54 Get NC-SI Pass-through Statistics response (0x9A)

In the absence of any error, the channel shall process and respond to the Get NC-SI Pass-through Statistics command by sending the response packet and payload shown in Table 106.

**Table 106 – Get NC-SI Pass-through Statistics response packet format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..27** | Pass-through TX Packets Received on NC-SI Interface<br>(Management Controller to Network Controller) | | | |
| **28..31** | Pass-through TX Packets Dropped | | | |
| **32..35** | Pass-through TX Packet Channel State Errors | | | |
| **36..39** | Pass-through TX Packet Undersized Errors | | | |
| **40..43** | Pass-through TX Packet Oversized Errors | | | |
| **44..47** | Pass-through RX Packets Received on LAN Interface | | | |
| **48..51** | Total Pass-through RX Packets Dropped | | | |
| **52..55** | Pass-through RX Packet Channel State Errors | | | |
| **56..59** | Pass-through RX Packet Undersized Errors | | | |
| **60..63** | Pass-through RX Packet Oversized Errors | | | |
| **64..67** | Checksum | | | |

The Get NC-SI Statistics Response frame contains a set of statistics counters that monitor the NC-SI Pass-through traffic in the Network Controller. Supported counters shall be reset to 0x0 when entering the Initial State and after being read. Implementation of the counters shown in Table 107 is optional. The Network Controller shall return any unsupported counter with a value of 0xFFFFFFFF for 32-bit counters and 0xFFFFFFFFFFFFFFFF for 64-bit counters. Counters may wraparound or stop if they reach 0xFFFFFFFE for 32-bit counters and 0xFFFFFFFFFFFFFFFE for 64-bit counters.

**Table 107 – Get NC-SI Pass-through Statistics counters**

| **Counter Number** | **Name** | **Meaning** |
|---|---|---|
| 1 | Total Pass-through TX Packets Received (Management Controller to Channel) | Counts the number of Pass-through packets forwarded by the channel to the LAN |
| 2 | Total Pass-through TX Packets Dropped (Management Controller to Channel) | Counts the number of Pass-through packets from the Management Controller that were dropped by the Network Controller |
| 3 | Pass-through TX Packet Channel State Errors (Management Controller to Channel) | Counts the number of egress management packets (Management Controller to Network Controller) that were dropped because the channel was in the disabled state when the packet was received |

| Counter Number | Name | Meaning |
|---|---|---|
| 4 | Pass-through TX Packet Undersized Errors (Management Controller to Channel) | Counts the number of Pass-through packets from the Management Controller that were undersized (under 64 bytes, including FCS) |
| 5 | Pass-through TX Packet Oversized Errors (Management Controller to Channel) | Counts the number of Pass-through packets from the Management Controller that were oversized (over 1522 bytes, including FCS) |
| 6 | Total Pass-through RX Packets Received on the LAN Interface (LAN to Channel) | Counts the number of Pass-through packets that were received on the LAN interface of the channel. This counter does not necessarily count the number of packets that were transmitted to the Management Controller, because some of the packets might have been dropped due to RX queue overflow. |
| 7 | Total Pass-through RX Packets Dropped (LAN to Channel) | Counts the number of Pass-through packets that were received on the LAN interface of the channel but were dropped and not transmitted to the Management Controller |
| 8 | Pass-through RX Packet Channel State Errors (LAN to Channel) | Counts the number of ingress management packets (channel to Management Controller) that were dropped because the channel was in the disabled state when the packet was received. The NC may also count packets that were dropped because the package was in the deselected state. |
| 9 | Pass-through RX Packet Undersized Errors (LAN to Channel) | Counts the number of Pass-through packets from the LAN that were undersized (under 64 bytes, including FCS) |
| 10 | Pass-through RX Packet Oversized Errors (LAN to Channel) | Counts the number of Pass-through packets from the LAN that were oversized (over 1522 bytes, including FCS) |

Currently no command-specific reason code is identified for this response.

### 8.4.55 Get Package Status command (`0x1B`)

The Get Package Status command provides a way for a Management Controller to explicitly query the status of a package. The Get Package Status command is addressed to the package, rather than to a particular channel (that is, the command is sent with a Channel ID where the Package ID subfield matches the ID of the intended package, and the Internal Channel ID subfield is set to `0x1F`).

Table 108 illustrates the packet format of the Get Package Status command.

**Table 108 – Get Package Status packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.4.56 Get Package Status response (0x9B)

In the absence of any errors, the package shall process and respond to the Get Package Status Command and send the response packet shown in Table 109.

Currently no command-specific reason code is identified for this response.

**Table 109 – Get Package Status response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Package Status | | | |
| **24..27** | Checksum | | | |
| **28..45** | Pad | | | |

**Table 110 – Package Status field bit definitions**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | Hardware Arbitration Status | 0b = Hardware arbitration is non-operational (inactive) or unsupported.<br>NOTE: This means that hardware arbitration tokens are not flowing through this NC.<br>1b = Hardware arbitration is supported, active, and implemented for the package on the given system. |
| 1 | Delayed Response Status | 0b = Delayed Response handling is disabled.<br>1b = Delayed Response handling is enabled. |
| 31.. 2 | Reserved | Reserved |

### 8.4.57 Get NC Capabilities and Settings command (0x25)

The Get NC Capabilities and Settings command is sent only as a package command. It is used to discover the supported architectural and currently configured (active) parameters of the NC.

Table 111 illustrates the packet format for the Get NC Capabilities and Settings command.

**Table 111 – Get NC Capabilities and Settings command packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

### 8.4.58 Get NC Capabilities and Settings response (`0xA5`)

In the absence of any errors, the package shall process and respond to the Get NC Capabilities and Settings Command and send the response packet shown in Table 112.

Currently no command-specific reason code is identified for this response.

**Table 112 - Get NC Capabilities and Settings response packet format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Max Ports | Enabled Ports | Max PCIe Endpoints | Enabled PCIe Endpoints |
| **24..27** | Max PFs | Enabled PFs | Max VFs | |
| **28..31** | Fabrics | Enabled Fabrics | Other Capabilities | |
| **32..35** | Checksum | | | |
| **36..45** | Pad | | | |

#### 8.4.58.1 Max Ports field

The Max Ports field indicates the maximum number of network ports that can be supported by the implementation (uint8).

#### 8.4.58.2 Enabled Ports field

The Enabled Ports field indicates the current number of network ports that are currently configured (uint8).

#### 8.4.58.3 Max PCIe Endpoints field

The Max PCIe Endpoints field indicates the maximum number of PCIe Endpoints that can be supported by the implementation (uint8).

#### 8.4.58.4 Enabled PCIe Endpoints field

The Enabled PCIe Endpoints field indicates the current number of PCIe Endpoints that are currently configured (uint8).

#### 8.4.58.5 Max PFs field

The Max PFs field indicates the maximum number of PCIe Physical Functions that can be supported by the implementation (uint8).

#### 8.4.58.6 Enabled PFs field

The Enabled PFs field indicates the current number of PCIe Physical Functions that are currently configured (uint8).

### 8.4.58.7 Max VFs field

The Max VFs field indicates the maximum number of PCIe Virtual Functions that can be supported by the implementation (uint8).

### 8.4.58.8 Fabrics field

The Fabrics field indicates the network fabrics that can be supported by the implementation.

**Table 113 – Fabrics field bit definitions**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | Ethernet | 0b0 = Ethernet Fabric is not supported<br>0b1 = Ethernet Fabric is supported |
| 1 | Fibre Channel | 0b0 = Fibre Channel Fabric is not supported<br>0b1 = Fibre Channel Fabric is supported |
| 2 | InfiniBand | 0b0 = InfiniBand Fabric is not supported<br>0b1 = InfiniBand Fabric is supported |
| 3..7 | Reserved | Reserved |

### 8.4.58.9 Enabled Fabrics field

The Enabled Fabrics field indicates the currently configured fabrics.

**Table 114 – Enabled Fabrics field bit definitions**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | Ethernet | 0b0 = Ethernet Fabric is not enabled<br>0b1 = Ethernet Fabric is enabled |
| 1 | Fibre Channel | 0b0 = Fibre Channel Fabric is not enabled<br>0b1 = Fibre Channel Fabric is enabled |
| 2 | InfiniBand | 0b0 = InfiniBand Fabric is not enabled<br>0b1 = InfiniBand Fabric is enabled |
| 3..7 | Reserved | Reserved |

### 8.4.58.10 Other Capabilities field

The Other Capabilities field indicates which features of this specification the NC supports, as described in Table 115.

**Table 115 – Capabilities Flags bit definitions**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | VF allocation | 0b = The Max VFs field is interpreted as per port<br>1b = The Max VFs field is interpreted as per device |
| 1 | Enabled Ports | 0b = The number of Enabled Ports is fixed<br>1b = The number of Enabled Ports is programmable |

| Bit Position | Field Description | Value Description |
|---|---|---|
| 2 | Enabled PCIe Endpoints | `0b` = The number of Enabled PCIe Endpoints is fixed<br>`1b` = The number of Enabled PCIe Endpoints is programmable |
| 3 | Enabled PFs | `0b` = The number of Enabled PFs is fixed<br>`1b` = The number of Enabled PFs is programmable |
| 4..8 | Max Data Transfer Size Exp | This value advertises the maximum value of data transfer length for the chunked data transfer. This field represents the 2s exponent of the maximum data transfer size.<br>`0x0` = Chunk Data Transfer is not supported.<br>`0x1` = Reserved.<br>`0x2` = Max Data Transfer Length supported is 4 (22) Bytes.<br>….<br>`0x8` = Max Data Transfer Length supported is 256 (28) Bytes.<br>….<br>`0x17` = Max Data Transfer Length supported is 8M (2^23) Bytes.<br>….<br>`0x1F` = Max Data Transfer Length supported is 2G (2^31) Bytes. |
| 9..15 | Reserved | Reserved |

### 8.4.59 Set NC Configuration command (`0x26`)

The Set NC Configuration command allows the Management Controller to configure the number of active Physical functions and PCIe (host) and network interfaces, where allowed (generally if the reported max value of the respective entity is greater than one). The values (programmed or fixed) are used in the PF Assignment command where the associations are made between the physical ports, partitions and host buses. If the implementation or controller architecture does not allow any configuration of these parameters, this command shall not be implemented.

The values configured by this command are held by the NC and only take effect at the next PCIe reset.

The Set NC Configuration command is addressed to the package, rather than to a channel (that is, the command is sent with a Channel ID where the Package ID subfield matches the ID of the intended package and the Internal Channel ID subfield is set to `0x1F`).

Table 116 illustrates the packet format of the Set NC Configuration command.

**Table 116 – Set NC Configuration command packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Enable Ports | Enable PCIe Endpoints | Enable PFs | Reserved |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

#### 8.4.59.1 Enable Ports field

The Enable Ports field (uint8) indicates the number of network ports to be enabled at the next PCIe reset.

#### 8.4.59.2 Enable PCIe Endpoints field

The Enable PCIe Endpoints field (uint8) indicates the number of PCIe Endpoints to be enabled at the next PCIe reset. In some implementation architectures this is not settable by NC-SI; in those cases this field becomes read-only and the value is ignored. PCIe Endpoint 0 shall be used if the Controller is configured for single bus operation.

#### 8.4.59.3 Enable PFs field

The Enable PFs field (uint8) indicates the number of PCIe Physical Functions to be enabled at the next PCIe reset.

### 8.4.60 Set NC Configuration response (0xA6)

The package shall, in the absence of a checksum error or identifier mismatch, always accept the Set NC Configuration command and send a response (see Table 117).

**Table 117 – Set NC Configuration response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.4.61 Get PF Assignment command (0x27)

The Get PF Assignment command is a Package command that allows the Management controller to receive the list of PCI Physical Functions (partitions) currently assigned to channels in the package, their enablement state and conditionally what PCIe Endpoint they are assigned to if the NC supports multiple host interfaces.

See the Set PF Assignment command description for additional information.

Table 118 illustrates the packet format of the Get PF Assignment Command.

**Table 118 – Get PF Assignment Command Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

## 8.4.62 Get PF Assignment Response `(0xA7)`

In the absence of any errors, the channel shall process and respond to the Get PF Assignment Command and send the response packet shown in the table below.

NOTE: Braces {} denote fields that depend on device capabilities.

**Table 119 – Get PF Assignment Response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Channel 0 Function Assignment bitmap | | | |
| **24..27** | {Channel 1 Function Assignment bitmap} | | | |
| | … | | | |
| | {Channel *c*-1 Function Assignment bitmap} | | | |
| | Function - Port Association | | | |
| | Function Enablement bitmap | | | |
| | {PCIe Endpoint 0 Function Assignment bitmap} | | | |
| | {PCIe Endpoint 1 Function Assignment bitmap} | | | |
| | … | | | |
| | {PCIe Endpoint *b*-1 Function Assignment bitmap} | | | |
| | Checksum | | | |
| | Pad | | | |

### 8.4.62.1 Channel Function Assignment bitmap fields

The number of Channel Function Assignment bitmaps returned in the response is equal to ‘c’, the number returned in the Enabled Ports field of Get NC Capabilities and Settings Response. The Channel Function Assignment bitmaps are 32-bit fields in which each bit position corresponds to a PCIe physical function in the NC on the specified channel. If a physical function is assigned to a channel, even if the physical function is not currently enabled, the value of the corresponding bit in the channel’s bitmap shall be set to `1b`; otherwise, the value shall be set to `0b`.

**Table 120 – Channel Function Assignment bitmap field**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | F0 status | `0b` = F0 is not assigned to the channel.<br>`1b` = F0 is assigned to the channel. |
| 1 | F1 status | `0b` = F1 is not assigned to the channel.<br>`1b` = F1 is assigned to the channel. |
| … | … | … |
| 31 | F31 status | `0b` = F31 is not assigned to the channel.<br>`1b` = F31 is assigned to the channel |

### 8.4.62.2 Function Port Association bitmap field

The Function Port Association bitmap is a 32-bit field in which each bit position corresponds to a physical function in the device.

**Table 121 – Function Port Association bitmap field**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | F0 association | `0b` = F0 is fixed to the specified channel.<br>`1b` = F0 may be assigned to any channel. |
| 1 | F1 association | `0b` = F1 is fixed to the specified channel.<br>`1b` = F1 may be assigned to any channel. |
| … | … | … |
| 31 | F31 association | `0b` = F31 is fixed to the specified channel.<br>`1b` = F31 may be assigned to any channel. |

### 8.4.62.3 Function Enablement bitmap field

The Function Assignment bitmap is a 32-bit field in which each bit position corresponds to a physical function in the NC. The number of functions shown as enabled in this field shall be equal to the value of Enabled PFs field in the Get NC Capabilities and Settings command. A function may be assigned to a PCIe Endpoint and be enabled and not be assigned to a channel in some implementations (i.e., a non-networking function).

**Table 122 – Function Enablement bitmap field**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | F0 status | `0b` = F0 is not enabled<br>`1b` = F0 is enabled |
| 1 | F1 status | `0b` = F1 is not enabled.<br>`1b` = F1 is enabled. |
| … | … | … |

| Bit Position | Field Description | Value Description |
|---|---|---|
| 31 | F31 status | 0b = F31 is not enabled.<br>1b = F31 is enabled |

#### 8.4.62.4 PCIe Endpoint Assignment bitmap field

The number of PCIe Endpoint Assignment bitmaps returned in the response is equal to 'b', the number returned in the Enabled PCIe Endpoints field of Get NC Capabilities and Settings response. The PCIe Endpoint b Assignment bitmaps are 32-bit fields in which each bit position corresponds to a physical function in the NC on the specified host bus. If the physical function is assigned to an PCIe Endpoint, even if the physical function is not currently enabled, the value of the corresponding bit shall be set to 1b, otherwise the value shall be set to 0b.

**Table 123 – PCIe Endpoint Assignment bitmap field**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | F0 status | 0b = F0 is not assigned on the specified PCIe Endpoint.<br>1b = F0 is assigned on the specified PCIe Endpoint. |
| 1 | F1 status | 0b = F1 is not assigned on the specified PCIe Endpoint.<br>1b = F1 is assigned on the specified PCIe Endpoint. |
| … | … | … |
| 31 | F31 status | 0b = F31 is not assigned on the specified PCIe Endpoint.<br>1b = F31 is assigned on the specified PCIe Endpoint |

#### 8.4.62.5 Calculation of Partition ID

When multiple functions are assigned to a channel, they are addressed by a value called the Partition ID. The Partition ID is created by taking the set of Functions that are assigned to a channel and assigning each an index value starting with the lowest numbered Function. A Function assigned to a channel has a Partition ID even if it is not enabled. Partition numbering starts at 0. For example, if F2 and F6 are assigned to channel 3, but only F2 is enabled, then F2 has Partition ID = 0 and F6 has Partition ID = 1 on that channel.

### 8.4.63 Set PF Assignment command (0x28)

The Set PF Assignment command is a Package command that allows the Management controller to enable, disable, and assign PCIe Physical Functions (partitions) in the controller to the channels, and, if applicable, to different PCI Endpoints in multi-home or multi-host configurations.

The format of the command payload is dependent on the numbers of Physical Functions, Channels and PCI Endpoints supported by the controller:

1) The number of Function Assignments bitmap fields shall be determined by the value (c) of the Enabled Ports field of Get NC Capabilities and Settings Response

2) The number of Physical Functions allowed to be configured in the Function Assignment and Enablement bitmap fields shall be determined by the value of the Enabled PFs field in the Get NC Capabilities and Settings command response. Assignment in all bitmaps starts at bit 0 and continues sequentially for the number of Functions supported. To support various

implementation architectures, the definition of assignment/enablement rules is beyond the scope of this specification.

3) If the value (b) of the Enabled PCIe Endpoints field of the Get NC Capabilities and Settings response is greater than 1, the Controller shall also include that number of PCIe Endpoint Function Assignment bitmaps in the command. Controllers that do not support multiple PCIe interfaces shall not implement PCIe Endpoint Host Function Assignment bitmap fields.

The values configured by this command are held by the controller and only take effect at the next PCIe reset. The configuration is persistent unless changed by another Set PF Assignment command or other mechanism.

Table 124 illustrates the packet format of the Set PF Assignment Command.

NOTE: Braces {} denote fields that depend on device capabilities.

**Table 124 – Set PF Assignment Command packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Channel 0 Function Assignment bitmap | | | |
| | {Channel 1 Function Assignment bitmap} | | | |
| | … | | | |
| | {Channel *c*-1 Function Assignment bitmap} | | | |
| | Function Enablement bitmap | | | |
| | {PCIe Endpoint 0 Function Assignment bitmap} | | | |
| | {PCIe Endpoint 1 Function Assignment bitmap} | | | |
| | … | | | |
| | {PCIe Endpoint *b*-1 Function Assignment bitmap} | | | |
| | Checksum | | | |
| | Pad | | | |

### 8.4.63.1 Channel Function Assignment bitmap field

The Channel Function Assignment bitmap is a 32-bit field in which each bit position corresponds to a physical function in the device. If the physical function is assigned to the channel, even if the physical function is not currently enabled, the corresponding bit value shall be set to 1b. This allows for a partition ID to be assigned and partition commands to be sent to the function even if it is not enabled.

**Table 125 – Channel Function Assignment bitmap field**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | F0 status | 0b = F0 is not assigned on the channel.<br>1b = F0 is assigned on the channel. |
| 1 | F1 status | 0b = F1 is not assigned on the channel.<br>1b = F1 is assigned on the channel. |

| Bit Position | Field Description | Value Description |
|---|---|---|
| … | … | … |
| 31 | F31 status | 0b = F31 is not assigned on the channel.<br>1b = F31 is assigned on the channel |

#### 8.4.63.2 Function Enablement bitmap field

The Function Assignment bitmap is a 32-bit field in which each bit position corresponds to a physical function in the device.

**Table 126 – Function Enablement bitmap field**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | F0 status | 0b = F0 is not enabled on the specified channel.<br>1b = F0 is enabled on the specified channel. |
| 1 | F1 status | 0b = F1 is not enabled on the specified channel.<br>1b = F1 is enabled on the specified channel. |
| … | … | … |
| 31 | F31 status | 0b = F31 is not enabled on the specified channel.<br>1b = F31 is enabled on the specified channel |

#### 8.4.63.3 PCIe Endpoint Assignment bitmap field

The PCIe Endpoint Assignment bitmap is a 32-bit field in which each bit position corresponds to a physical function in the device.

**Table 127 – PCIe Endpoint Assignment bitmap field**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | F0 status | 0b = F0 is not assigned on the specified PCIe Endpoint.<br>1b = F0 is assigned on the specified PCIe Endpoint. |
| 1 | F1 status | 0b = F1 is not assigned on the specified PCIe Endpoint.<br>1b = F1 is assigned on the specified PCIe Endpoint. |
| … | … | … |
| 31 | F31 status | 0b = F31 is not assigned on the specified PCIe Endpoint.<br>1b = F31 is assigned on the specified PCIe Endpoint |

### 8.4.64 Set PF Assignment Response (0xA8)

In the absence of any errors, the channel shall process and respond to the Set PF Assignment Command and send the response packet shown in Table 128.

**Table 128 – Set PF Assignment Response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **24..27** | Checksum | | | |
| **36..39** | Pad | | | |

### 8.4.65 Get Channel Configuration command (0x29)

The Get Channel Configuration command is used to discover the currently configured settings of the channel, including the fabric type, the implemented media type, the number of enabled partitions, if any, and their bandwidth allocation settings where applicable.

Table 129 illustrates the packet format for the Get Channel Configuration command.

**Table 129 – Get Channel Configuration command packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

### 8.4.66 Get Channel Configuration response (0xA9)

In the absence of any errors, the channel shall process and respond to the Get Channel Configuration Command and send the response packet shown in Table 130.

Currently no command-specific reason code is identified for this response.

**Table 130 – Get Channel Configuration response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Fabric Type | Media Type | Max MTU | |
| **24..27** | Reserved | | Reserved | Num Enabled Partitions |
| **28..31** | P1 Max TX BW | P1 Min TX BW | P2 Max TX BW | P2 Min TX BW |
| **…** | …. | | | |
| | Checksum | | | |
| | Pad | | | |

#### 8.4.66.1 Fabric Type field

The Fabric Type field indicates which personality types are currently enabled on the channel, as described in Table 131.

Table 131 – Fabric Type definitions

| Value | Fabric Type | Value Description |
|---|---|---|
| 1 | Ethernet Mode | Ethernet operation is enabled |
| 2 | Fibre Channel Mode | Fibre Channel operation is enabled |
| 3 | InfiniBand Mode | InfiniBand operation is enabled |
| All others | Reserved | Reserved |

#### 8.4.66.2 Media Type field

The Media Type field indicates the physical interface type used on the port implementation and if that port supports one or more than one NC-SI channels (for example, some designs may support up to 4 independent ports in a QSFP interface), as described in Table 132.

NOTE: An implementation that implements a SFF cage interface into which a RJ-45 transceiver is plugged shall return ‘SFF cage’ as the media type.

Table 132 – Media Type bit definitions

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | Backplane | `0b` = The media does not have a backplane interface<br>`1b` = The media has a backplane interface |
| 1 | Base-T (RJ-45 style) | `0b` = The media does not have a Base-T interface<br>`1b` = The media has a Base-T (RJ-45 style) interface |
| 2 | SFF cage | `0b` = The media does not have an SFF-style interface<br>`1b` = The media has an SFF-style interface |
| 3..6 | Reserved | Reserved |
| 7 | Shared Interface | 0b = The media is dedicated to one NC-SI channel<br>1b = The media is shared between multiple channels |

#### 8.4.66.3 Max MTU field

The Max MTU field is used to report the maximum allowed MTU size (Bytes) when the port is configured for Ethernet.

#### 8.4.66.4 Num Enabled Partitions

The Num Enabled Partitions field indicates the number of Functions that have been assigned to the channel/port. This field is used only to provide the number of partitions present in the bandwidth fields. If the Num Enabled Partitions is an odd number, then the last two bytes before the Checksum field shall be reserved and set to 0.

#### 8.4.66.5 P(n) Max TX BW Fields

These fields contain the Maximum TX bandwidth allocation of the $n^{th}$ enabled partition expressed in % of the physical port link speed.

#### 8.4.66.6 P(n) Min TX BW Fields

These fields contain the Minimum TX bandwidth allocation of the nth enabled partition expressed in % of the physical port link speed.

### 8.4.67 Set Channel Configuration command (`0x2A`)

The Set Channel Configuration command allows the Management Controller to configure characteristics of the channel. The TX Bandwidth fields must be set for each enabled partition, but their values may be overridden during operation by other configuration methods (outside of the scope of this specification)’

Table 133 illustrates the packet format of the Set Channel Configuration command.

**Table 133 – Set Channel Configuration command packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Fabric Type | Num Partitions | Max MTU | |
| **20..23** | P1 Max TX BW | P1 Min TX BW | P2 Max TX BW | P2 Min TX BW |
| | … | | | |
| | Checksum | | | |
| | Pad | | | |

#### 8.4.67.1 Fabric Type field

The Fabric Type field indicates the personality type to be enabled on the channel, as described in Table 134. The contents of this field may be ignored if the channel only supports one fabric type. The Fabric type is a channel property shared by all partitions assigned to the channel.

**Table 134 – Fabric Type definitions**

| Value | Fabric Type | Value Description |
|---|---|---|
| 1 | Ethernet Mode | Enable Ethernet operation |
| 2 | Fibre Channel Mode | Enable Fibre Channel operation |
| 3 | InfiniBand Mode | Enable InfiniBand operation |
| all others | Reserved | Reserved |

#### 8.4.67.2 Max MTU field

The Max MTU field is used to configure the maximum allowed MTU size (Bytes) when the port is configured for Ethernet or InfiniBand.

#### 8.4.67.3 Num Partitions

The Num Partitions field indicates the number of Functions that have been assigned to the channel/port in the Set PF Assignment command. This field is used only to provide the number of partitions present in the bandwidth fields and does not have the ability to change the number of assigned partitions on the channel. Each assigned partition must be allocated min and max TX bandwidth values when enabled.

The initial value is generally expected to be one partition enabled per port and if modified, the new value should persist across system boot and power cycles.

If the Num Partitions is an odd number, then the last two bytes before the Checksum field shall be reserved and set to 0.

#### 8.4.67.4 P(n) Max TX BW fields

These fields contain the Maximum TX bandwidth allocation of the nth enabled partition expressed in % of the physical port link speed. Oversubscription of partition maximum bandwidth is allowed. The field value is an integer ranging from 0 to $100_{10}$.

The initial value is generally expected to be 100% per partition, allowing each enabled partition full use of the channel bandwidth if no other partition has traffic. If modified, the new value should persist across system boot and power cycles.

#### 8.4.67.5 P(n) Min TX BW field

These fields contain the Minimum TX bandwidth allocation of the nth enabled partition expressed in % of the physical port link speed. This is interpreted as committed bandwidth to the partition and as such the Min TX BW fields of all enabled partitions on the port must sum to 100%. The field value is an integer ranging from 0 to $100_{10}$.

The initial value is generally expected to be equal weighting among all enabled partitions, allowing each enabled partition equal use of the channel bandwidth. If modified, the new value should persist across system boot and power cycles.

### 8.4.68 Set Channel Configuration response (0xAA)

The package shall, in the absence of a checksum error or identifier mismatch, always accept the Set Channel Configuration command and send a response (see Table 135).

**Table 135 – Set Channel Configuration response packet format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.4.69 Get Partition Configuration command (`0x2B`)

The Get Partition Configuration command is used to discover additional optional functions supported by the channel, such as the number of unicast/multicast addresses supported.

Table 136 illustrates the packet format for the Get Partition Configuration command.

**Table 136 – Get Partition Configuration command packet format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Partition ID | Reserved | | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

#### 8.4.69.1 Partition ID field

The Partition ID field is the identifier for the function on the channel as defined in clause 8.4.62.5.

### 8.4.70 Get Partition Configuration response (`0xAB`)

In the absence of any errors, the channel shall process and respond to the Get Partition Configuration Command and send the response packet shown in Table 137.

Currently no command-specific reason code is identified for this response.

**Table 137 – Get Partition Configuration response packet format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Personality Cfg | Personality Spt | Configuration Flags | |
| **24..27** | Max TX BW | Min TX BW | Reserved | |
| **28..31** | PCI DID | | PCI VID | |
| **32..35** | PCI SSID | | PCI SVID | |
| **36..39** | PCIe Endpoint # | PCIe Bus # | PCIe Device # | PCIe Function # |
| **40..43** | Reserved | Address Count | Address TLVs… | |
| | … | … | … | … |
| | Checksum | | | |

#### 8.4.70.1 Personality Cfg field

The Personality Configured field indicates which personality type(s) are currently enabled on the partition, as described in Table 138.

NOTE: Some implementations may support multiple personalities being simultaneously enabled.

**Table 138 – Personality Cfg bit definitions**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | Ethernet Status | 0b = Ethernet operation is not enabled<br>1b = Ethernet operation is enabled |
| 1 | Fibre Channel Status | 0b = Fibre Channel operation is not enabled<br>1b = Fibre Channel operation is enabled |
| 2 | Fibre Channel over Ethernet Status | 0b = Fibre Channel over Ethernet operation is not enabled<br>1b = Fibre Channel over Ethernet operation is enabled |
| 3 | InfiniBand Status | 0b = InfiniBand operation is not enabled<br>1b = InfiniBand operation is enabled |
| 4 | iSCSI Offload Status | 0b = iSCSI Offload operation is not enabled<br>1b = iSCSI Offload operation is enabled |
| 5 | RDMA Status | 0b = RDMA operation is not enabled<br>1b = RDMA operation is enabled |
| 6 | NVMe | 0b = NVMe operation is not enabled<br>1b = NVMe operation is enabled |
| 7 | Reserved | Reserved |

#### 8.4.70.2 Personality Spt field

The Personality Supported field indicates which personality types the partition supports, as described in Table 139.

**Table 139 – Personality Spt bit definitions**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | Ethernet Support | 0b = Ethernet operation is not supported<br>1b = Ethernet operation is supported |
| 1 | Fibre Channel Support | 0b = Fibre Channel operation is not supported<br>1b = Fibre Channel operation is supported |
| 2 | Fibre Channel over Ethernet Support | 0b = Fibre Channel over Ethernet operation isn’t supported<br>1b = Fibre Channel over Ethernet operation is supported |
| 3 | InfiniBand Support | 0b = InfiniBand operation is not supported<br>1b = InfiniBand operation is supported |
| 4 | iSCSI Offload Support | 0b = iSCSI Offload operation is not supported<br>1b = iSCSI Offload operation is supported |
| 5 | RDMA Support | 0b = RDMA operation is not supported<br>1b = RDMA operation is supported |
| 6 | NVMe | 0b = NVMe Offload operation is not supported<br>1b = NVMe Offload operation is supported |
| 7 | Reserved | Reserved |

#### 8.4.70.3 Configuration Flags field

The Configuration Flags field indicates which optional features of this specification the channel supports, as described in Table 140.

**Table 140 – Configuration Flags bit definitions**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | Host Driver Status | `0b` = When reporting is supported, a Host driver is not present on the partition<br>`1b` = When reporting is supported, a Host driver is present on the partition |
| 1 | Host Driver Status Reporting | `0b` = Host Driver status reporting is not supported.<br>`1b` = Host Driver status reporting (bit 0) is supported. |
| 2..3 | Partition Link Status | `00b` = When reporting is supported, Partition Link is down<br>`01b` = When reporting is supported, Partition Link is forced up<br>`10b` = When reporting is supported, Partition Link follows Channel Link<br>`11b` = Reserved |
| 4 | Partition Link Status Reporting | `0b` = Partition Link Status reporting is not supported.<br>`1b` = Partition Link Status reporting (bit 2) is supported. |
| 5 | Boot Status | `0b` = The partition is not configured for boot.<br>`1b` = The partition is configured for boot. |
| 6 | Bootable | `0b` = The partition does not support boot<br>`1b` = The partition supports boot and reporting |
| 7..31 | Reserved | Reserved |

#### 8.4.70.4 Max TX BW field

This field contains the Maximum TX bandwidth allocation of the partition expressed in % of the physical port link speed. The % value ranges from 0 to $100_{10}$ represented as an integer.

#### 8.4.70.5 Min TX BW field

This field contains the Minimum TX bandwidth allocation of the partition expressed in % of the physical port link speed. This is interpreted as committed bandwidth to the partition and as such the Min TX BW fields of all enabled partitions on the port must sum to 100%. The % value ranges from 0 to $100_{10}$ represented as an integer.

#### 8.4.70.6 PCI DID

The current PCI Device ID of the Partition.

#### 8.4.70.7 PCI VID

The current PCI Vendor ID of the Partition.

#### 8.4.70.8 PCI SSID

The current PCI Subsystem ID of the Partition.

#### 8.4.70.9 PCI SVID

The current PCI Subvendor ID of the Partition.

#### 8.4.70.10 PCIe Endpoint #

The identifier indicating which PCIe Endpoint on the NC the partition is associated with.

#### 8.4.70.11 PCIe Bus #

The assigned primary PCIe Bus number assigned to the partition in the host system's bus enumeration process. If PCIe Device number is set to 0xFE, then PCIe Bus number field shall be ignored.

#### 8.4.70.12 PCIe Device #

The assigned PCIe Device number assigned to the partition except in the cases of ARI mode operation when it shall contain the value of 0xFF. If the PCIe Device number is set to 0xFE, then PCIe Bus #, PCIe Device #, and PCIe Function # fields shall be ignored and the partition shall be considered a partition with an unassigned routing ID (Bus, Device, Function number or Bus, Function number).

#### 8.4.70.13 PCIe Function #

The assigned PCIe Function number assigned to the partition in the host system's bus enumeration process. If the PCIe Device number is set to any value between 0x00 to 0x1f, then the PCIe function number shall be a value between 0x00 to 0x07. If PCIe Device number is set to 0xFE, then PCIe Function number field shall be ignored.

#### 8.4.70.14 Address Count field

This field indicates the number of permanent and virtual addresses reported by the partition.

#### 8.4.70.15 Address TLVs

These TLVs show the permanently programmed and current addresses being used by the partition.

**Table 141 – Address Type-Length-Value Field Bit Definitions**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 7..0 | Address Type | The following type encodings shall be used to indicate the address values that are permanently assigned to the partition. The response shall include all types whether or not that mode of operation is active, or the partition is enabled:<br>0x0 = Reserved<br>0x1 = Ethernet MAC<br>0x2 = iSCSI Offload (Ethernet MAC)<br>0x3 = Fibre Channel World Wide Node Name<br>0x4 = Fibre Channel World Wide Port Name<br>0x5 = FCoE-FIP MAC |

| Bit Position | Field Description | Value Description |
|---|---|---|
| | | 0x6 = InfiniBand Node GUID<br>0x7 = InfiniBand Port GUID<br>0x8 = InfiniBand VPort/LID<br>The following type encodings shall be used to indicate all address values that are currently in use by the partition based on configured mode of operation. These may be the permanently assigned address or a programmatically assigned address.<br>:<br>0xF1 = Ethernet MAC<br>0xF2 = iSCSI Offload (Ethernet MAC)<br>0xF3 = Fibre Channel World Wide Node Name<br>0xF4 = Fibre Channel World Wide Port Name<br>0xF5 = FCoE-FIP MAC<br>0xF6 = InfiniBand Node GUID<br>0xF7 = InfiniBand Port GUID<br>0xF8 = InfiniBand VPort/LID<br>all others = Reserved |
| 15..8 | Address Length | The length indicates the number of bytes used in the address |
| … | Address | Address Length number of bytes of the Address |

### 8.4.71 Set Partition Configuration command (0x2C)

The Set Partition Configuration command allows the Management Controller to configure various settings of the partition including virtual addresses, VF allocation and other parameters.

The Set Partition Configuration command is addressed to the channel with the Partition ID field set to the index/ordinal of the target PF on the channel.

The partition’s personality configuration may be made persistent if written to the NVRAM via the Settings Commit command. These settings take effect at the next PCIe Reset.

Table 142 illustrates the packet format of the Set Partition Configuration command.

**Table 142 – Set Partition Configuration command packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Partition ID | Personality Cfg | Reserved | |
| **20..23** | Partition Link Control | Reserved | Address Count | Address TLV |
| **24..27** | Checksum | | | |
| **28..45** | Pad | | | |

### 8.4.71.1 Personality Cfg field

The Personality Configuration field indicates which personality type(s) shall be enabled on the partition, as described in Table 143. Any attempt to enable a personality not shown as supported in clause 8.4.70.2 shall be cause the command to fail with Parameter Is Invalid reason code. In some implementations it may be appropriate to select more than one personality at a time, for instance Ethernet and RDMA.

**Table 143 – Personality Cfg bit definitions**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | Ethernet Status | `0b` = Disable Ethernet operation<br>`1b` = Enable Ethernet operation |
| 1 | Fibre Channel Status | `0b` = Disable Fibre Channel operation<br>`1b` = Enable Fibre Channel operation |
| 2 | Fibre Channel over Ethernet Status | `0b` = Disable Fibre Channel over Ethernet operation<br>`1b` = Enable Fibre Channel over Ethernet operation |
| 3 | InfiniBand Status | `0b` = Disable InfiniBand operation<br>`1b` = Enable InfiniBand operation |
| 4 | iSCSI Offload Status | `0b` = Disable iSCSI Offload operation<br>`1b` = Enable iSCSI Offload operation |
| 5 | RDMA Status | `0b` = Disable RDMA operation<br>`1b` = Enable RDMA operation |
| 6 | NVMe | `0b` = Disable NVMe operation<br>`1b` = Enable NVMe operation |
| 7 | Reserved | Reserved |

#### 8.4.71.2 Partition Link Control

Table 144 describes the values for the Partition Link Control field.

**Table 144 – Values for the Partition Link Control field (8-bit field)**

| Value | Description |
|---|---|
| `0x0` | Partition Link is down |
| `0x1` | Partition Link is forced up |
| `0x2` | Partition Link follows Channel link state |
| `0x3..0xFF` | Reserved |

#### 8.4.71.3 Address Count field

The Address Count field contains the number of partition virtual addresses to be configured as specified in the Address TLV field.

#### 8.4.71.4 Address TLV

**Table 145 – Address Type-Length field bit definitions**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 7..0 | Address Type | Addresses specified herein override the permanent or factory-programmed network address to be used by the partition based on configured mode of operation. To return to using the permanent address, supply either an address of |

| Bit Position | Field Description | Value Description |
|---|---|---|
| | | 0 or the permanent address in this field or remove power from the NC.<br>:<br>0xF1 = Ethernet MAC<br>0xF2 = iSCSI Offload (Ethernet MAC)<br>0xF3 = Fibre Channel World Wide Node Name<br>0xF4 = Fibre Channel World Wide Port Name<br>0xF5 = FCoE-FIP MAC<br>0xF6 = InfiniBand Node GUID<br>0xF7 = InfiniBand Port GUID<br>0xF8 = InfiniBand VPort/LID<br>All others = Reserved |
| 15..8 | Address Length | The length indicates the number of bytes used in the address |

### 8.4.72 Set Partition Configuration response (0xAC)

The package shall, in the absence of a checksum error or identifier mismatch, always accept the Set Partition Configuration command and send a response (see Table 146).

**Table 146 – Set Partition Configuration response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.4.73 Get Boot Config Command (0x2D)

The Get Boot Config Command allows the Management Controller to query for the Boot Initiator settings of a given Boot Protocol type configured on the channel/PF/partition and stored in the NVRAM of the controller.

If the command is sent to a destination that exists but that does not support the specified Boot Protocol type, the command execution shall fail with a reason code indicating a Parameter Is Invalid, Unsupported, or Out-of-Range.

Table 147 illustrates the packet format of the Get Boot Config command.

**Table 147 – Get Boot Config command packet**

| | Bits |
|---|---|

| Bytes | 31..24 | 23..16 | 15..08 | 07..00 |
|---|---|---|---|---|
| 00..15 | NC-SI Header | | | |
| 16..19 | Partition ID | Reserved | Reserved | Protocol Type |
| 20..23 | Checksum | | | |
| 24..45 | Pad | | | |

#### 8.4.73.1 Protocol Type field

The Protocol Type field specifies the boot protocol for which configuration data is requested.

**Table 148 – Protocol Type field**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 7..0 | Boot Protocol Type | `0x0` = PXE<br>`0x1` = iSCSI<br>`0x2` = FCoE<br>`0x3` = FC<br>`0x4` = NVMeoFC<br>`0x5-0xFF` = Reserved |

### 8.4.74 Get Boot Config Response (`0xAD`)

The channel shall, in the absence of a checksum error or identifier mismatch, always accept the Get Boot Config command and send a response.

The Get Boot Config Response frame contains the currently stored settings for the specified Boot Protocol type contained in the controller's NVRAM that the channel/PF/partition will use in a boot operation done locally by the adapter. Settings that the Controller supports but does not have a value for (e.g., have no initial or current value) should be included in the Response and have a length of 0.

All string values specified in this command shall be in the unterminated ASCII string format.

Table 149 illustrates the packet format of the Get Boot Config Response.

**Table 149 – Get Boot Config Response packet**

| | Bits | | | |
|---|---|---|---|---|
| Bytes | 31..24 | 23..16 | 15..08 | 07..00 |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Reserved | | Protocol Type | Number of TLVs |
| 24.. | Type-Length Field #1 | | Value Field #1 | |
| … | Type-Length Field #2 | | Value Field #2 | |
| … | … | | | |

| | Bits | | | |
|---|---|---|---|---|
| Bytes | 31..24 | 23..16 | 15..08 | 07..00 |
| …. | Checksum | | | |

### 8.4.74.1 Protocol Type field

The Protocol Type field specifies the boot protocol for which boot attributes are being returned.

**Table 150 – Protocol Type field**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 7..0 | Boot Protocol Type | 0x0 = PXE<br>0x1 = iSCSI<br>0x2 = FCoE<br>0x3 = FC<br>0x4 = NVMeoFC<br>0x5-0xFE = Reserved<br>0xFF = Unknown protocol type |

### 8.4.74.2 Boot Protocol Type-Length-Value fields

The set of boot attributes (one of the following 4 tables) that correspond to the specified Protocol Type in the Command are returned as TLVs in the Response.

**Table 151 – PXE Boot Protocol Type-Length field**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 7..0 | Attribute Name/Type | 0x0 = VLAN ID (uint16)<br>0x1 = VLAN enable (bool8)<br>0x2-0xFF = Reserved |
| 15..8 | Length | |
| | Attribute Value | Value data |

3271 **Table 152 – Get FC Boot Protocol Type-Length field**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 7..0 | Attribute Name/Type | 0x0 = FCInitiatorBootSelection (uint8)<br>0x1 = FirstFCTargetWWPN (string)<br>0x2 = FirstFCTargetLUN (uint64)<br>0x3 = SecondFCTargetWWPN (string)<br>0x4 = SecondFCTargetLUN (uint64)<br>0x5 = ThirdFCTargetWWPN (string)<br>0x6 = ThirdFCTargetLUN (uint64)<br>0x7 = FourthFCTargetWWPN (string)<br>0x8 = FourthFCTargetLUN (uint64)<br>0x9 = FifthFCTargetWWPN (string)<br>0xA = FifthFCTargetLUN (uint64)<br>0xB = SixthFCTargetWWPN (string)<br>0xC = SixthFCTargetLUN (uint64)<br>0xD = SeventhFCTargetWWPN (string)<br>0xE = SeventhFCTargetLUN (uint64)<br>0xF = EighthFCTargetWWPN (string)<br>0x10 = EighthFCTargetLUN (uint64)<br>0x11-0xFF = Reserved |
| 15..8 | Length | |
| | Attribute Value | Value data |

3272

3273 **Table 153 – FCoE Boot Protocol Type-Length field**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 7..0 | Attribute Name/Type | 0x0 = FCoEInitiatorBootSelection<br>0x1 = FirstFCoEWWPNTarget (string)<br>0x2 = FirstFCoEBootTargetLUN (uint64)<br>0x3 = FirstFCoEFCFVLANID (uint16)<br>0x4 = FCoETgTBoot (bool8)<br>0x5-0xF = Reserved |
| 15..8 | Length | |
| | Attribute Value | Value data |

3274

### Table 154 – iSCSI Boot Protocol Type-Length field

| Bit Position | Field Description | Value Description |
|---|---|---|
| 7..0 | Attribute Name/Type | 0x0 = IscsiInitiatorIPAddrType (uint8)<br>0x1 = IscsiInitiatorAddr (string)<br>0x2 = IscsiInitiatorName (string)<br>0x3 = IscsiInitiatorSubnet (string)<br>0x4 = IscsiInitiatorSubnetPrefix (string)<br>0x5 = IscsiInitiatorGateway (string)<br>0x6 = IscsiInitiatorFirstDNS (string)<br>0x7 = IscsiInitiatorSecondDNS (string)<br><br>0x10 = ConnectFirstTgt (bool8)<br>0x11 = FirstTgtIpAddress (string)<br>0x12 = FirstTgtTcpPort (uint16)<br>0x13 = FirstTgtBootLun (uint64)<br>0x14 = FirstTgtIscsiName (string)<br>0x15 = FirstTgtChapId (string)<br>0x16 = FirstTgtChapPwd (string)<br>0x17 = FirstTgtVLANEnable (bool8)<br>0x18 = FirstTgtVLAN (uint16)<br><br>0x20 = ConnectSecondTgt (bool8)<br>0x21 = SecondTgtIpAddress (string)<br>0x22 = SecondTgtTcpPort (uint16)<br>0x23 = SecondTgtBootLun (uint64)<br>0x24 = SecondTgtIscsiName (string)<br>0x25 = SecondTgtChapId (string)<br>0x26 = SecondTgtChapPwd (string)<br>0x27 = SecondTgtVLANEnable (bool8)<br>0x28 = SecondTgtVLAN (uint16)<br><br>All others = Reserved |
| 15..8 | Length | |
| | Attribute Value | Value data |

**Table 155 – NVMeoFC Boot Protocol Type-Length field**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 7..0 | Attribute Name/Type | 0x0 = FirstNVMeTargetNQN (string)<br>0x1 = FirstNVMeTargetWWN (string)<br>0x2 = FirstNVMeTargetWWPN (string)<br>0x3 = FirstNVMeTgtConn (bool8)<br>0x4 = FirstNVMeTgtCntlrID (uint32)<br>0x5 = FirstNVMeTgtNSID (uint32)<br>0x6-0x7 = Reserved<br><br>0x8 = SecondNVMeTargetNQN (string)<br>0x9 = SecondNVMeTargetWWN (string)<br>0xA = SecondNVMeTargetWWPN (string)<br>0xB = SecondNVMeTgtConn (bool8)<br>0xC = SecondNVMeTgtCntlrID (uint32)<br>0xD = SecondNVMeTgtNSID (uint32)<br>0xE-0xF = Reserved<br><br>0x10 = ThirdNVMeTargetNQN (string)<br>0x11 = ThirdNVMeTargetWWN (string)<br>0x12 = ThirdNVMeTargetWWPN (string)<br>0x13 = ThirdNVMeTgtConn (bool8)<br>0x14 = ThirdNVMeTgtCntlrID (uint32)<br>0x15 = ThirdNVMeTgtNSID (uint32)<br>0x16-0x17 = Reserved<br><br>0x18 = FourthNVMeTargetNQN (string)<br>0x19 = FourthNVMeTargetWWN (string)<br>0x1A = FourthNVMeTargetWWPN (string)<br>0x1B = FourthNVMeTgtConn (bool8)<br>0x1C = FourthNVMeTgtCntlrID (uint32)<br>0x1D = FourthNVMeTgtNSID (uint32)<br>0x1E-0x1F = Reserved<br><br>0x20 = FifthNVMeTargetNQN (string)<br>0x21 = FifthNVMeTargetWWN (string)<br>0x22 = FifthNVMeTargetWWPN (string) |

| Bit Position | Field Description | Value Description |
|---|---|---|
| | | 0x23 = FifthNVMeTgtConn (bool8)<br>0x24 = FifthNVMeTgtCntlrID (uint32)<br>0x25 = FifthNVMeTgtNSID (uint32)<br>0x26-0x27 = Reserved<br><br>0x28 = SixthNVMeTargetNQN (string)<br>0x29 = SixthNVMeTargetWWN (string)<br>0x2A = SixthNVMeTargetWWPN (string)<br>0x2B = SixthNVMeTgtConn (bool8)<br>0x2C = SixthNVMeTgtCntlrID (uint32)<br>0x2D = SixthNVMeTgtNSID (uint32)<br>0x2E-0x2F = Reserved<br><br>0x30 = SeventhNVMeTargetNQN (string)<br>0x31 = SeventhNVMeTargetWWN (string)<br>0x32 = SeventhNVMeTargetWWPN (string)<br>0x33 = SeventhNVMeTgtConn (bool8)<br>0x34 = SeventhNVMeTgtCntlrID (uint32)<br>0x35 = SeventhNVMeTgtNSID (uint32)<br>0x36-0x37 = Reserved<br><br>0x38 = EighthNVMeTargetNQN (string)<br>0x39 = EighthNVMeTargetWWN (string)<br>0x3A = EighthNVMeTargetWWPN (string)<br>0x3B = EighthNVMeTgtConn (bool8)<br>0x3C = EighthNVMeTgtCntlrID (uint32)<br>0x3D = EighthNVMeTgtNSID (uint32)<br>0x3E-0xFF = Reserved |
| 15..8 | Length | |
| | Attribute Value | Value data |

### 8.4.75 Set Boot Config command (0x2E)

The Set Boot Config command allows the Management Controller to send to the channel/PF/partition the Boot settings to be used by the channel/PF/partition in conducting boot operations of the specified type.

The Network Controller shall apply the attribute values in the order received in this command (e.g., TLV1 before TLV2, etc.) so that any dependency relationships are maintained.

See the Get Boot Config Command for the definition of the **command** fields.

All string values specified in this command shall be in unterminated ASCII string format.

A NC that does not support or is not in partitioning mode shall have the Partition ID field programmed as `0x00`.

A TLV length value of 0 indicates the clearing of the current value of the attribute to null or no value.

A maximum of 32 TLVs may be sent in any one instance of the Set Boot Config command.

If the command is sent to a destination that exists but that does not support the specified Boot Protocol type, the command execution shall fail with a reason code of Parameter Is Invalid, Unsupported, or Out-of-Range.

**Table 156 – Set Boot Config command packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Partition ID | | Protocol Type | Number of TLVs |
| **24..** | Type-Length Field #1. | | Value Field #1. | |
| **….** | Type-Length Field #2 | | Value Field #2 | |
| **….** | **….** | | | |
| **….** | Checksum | | | |
| **….** | Pad | | | |

## 8.4.76 Set Boot Config Response (0xAE)

The channel shall, in the absence of a checksum error or identifier mismatch, always accept the Set Boot Config command and send a response.

Only if all the TLVs are accepted without error then the Command Completed/No Error response/reason code shall be returned with the TLV Error Reporting field set to all 0's.
If the command is sent to a destination that exists but that does not support the specified Boot Protocol type, the command response shall return the Parameter Is Invalid, Unsupported, or Out-of-Range reason code.

If there are errors in any of the TLVs included in the Set command, the entire command is deemed to fail, and no configuration changes are to be made by the controller. The TLV Error Reporting field shall be used to provide individual status reporting on the TLVs received.

**Table 157 – Set Boot Config Response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | TLV Error Reporting | | | |
| **28..31** | Checksum | | | |
| **32..45** | Pad | | | |

### 8.4.76.1 TLV Error Reporting field

The TLV Error Reporting field is a bitmap indicating which TLVs in the incoming Set command were processed without error, and which were not. The bit order corresponds to the order of TLVs in the incoming Set command as shown. There is a 1:1 correspondence between incoming TLVs and the active bits in this field. If fewer than 32 TLVs are transmitted, the bits corresponding to the unsent TLVs shall be set to 0.

**Table 158 – TLV Error Reporting field**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | TLV #1 status | `0b` = No error detected in TLV #1 or TLV #1 is not present<br>`1b` = Error detected in TLV #1 |
| … | … | … |
| 31 | TLV #32 status | `0b` = No error detected in TLV #32 or TLV #32 is not present<br>`1b` = Error detected in TLV #32 |

## 8.4.77 Get Partition Statistics command `(0x2F)`

The Get Partition Statistics command is used to retrieve network statistics relevant to the partition from the NC. For example, the MC should only request Ethernet statistics from a partition configured for Ethernet operation. The defined responses are customized for each personality type.

Implementation of this command is conditional and is required only for NCs that support partitioning. Implementation of each response type is conditional based on the NC supporting the specified type of operation on the partition.

The NC shall return in the response a value of `0xFFFFFFFF` for unsupported 32-bit counters and `0xFFFFFFFFFFFFFFFF` for unsupported 64-bit counters. For implementations that declare a particular counter only occupies 32 bits in a defined 64-bit (upper/lower) field, the lower field shall be used to provide the count and the upper field shall be set to `0xFFFFFFFF`.

As the intent of the command is to retrieve live statistics from enabled partitions, if the command is sent to a Partition ID that doesn't exist in the current configuration or if the Stats type does not match the configured personality of the partition, the command shall fail with the Parameter is Invalid reason code.

**Table 159 – Get Partition Statistics command packet format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Partition ID | Reserved | | Stats Type |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

#### 8.4.77.1 Stats Type field

The Stats Type field is the identifier for the type of statistics to be queried.

**Table 160 – Stats Type Field**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 7..0 | Stats Type | 0x01 = Ethernet<br>0x02 = iSCSI<br>0x04 = FCoE<br>0x08 = RDMA<br>0x10 = IB<br>0x20 = FC<br>All others = Reserved |

### 8.4.78 Get Partition Statistics response for Ethernet `(0xAF)`

In the absence of any errors, the channel shall process and respond to the Get Partition Statistics Command and send the response packet shown below when the Stats Type indicates Ethernet.

Currently no command-specific reason code is identified for this response.

**Table 161 – Get Partition Statistics (Ethernet) response packet format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Stats Type | Counter Sizes | Counters Cleared | |
| **24..27** | Total Bytes Received (upper) | | | |
| **28..31** | Total Bytes Received (lower) | | | |
| **32..35** | Total Bytes Transmitted (upper) | | | |
| **36..39** | Total Bytes Transmitted (lower) | | | |
| **40..43** | Total Unicast Packets Received | | | |
| **44..47** | Total Multicast Packets Received | | | |

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **48..51** | Total Broadcast Packets Received | | | |
| **52..55** | Total Unicast Packets Transmitted | | | |
| **56..59** | Total Multicast Packets Transmitted | | | |
| **60..63** | Total Broadcast Packets Transmitted | | | |
| **64..67** | Total Unicast Bytes Received (upper) | | | |
| **68..71** | Total Unicast Bytes Received (lower) | | | |
| **72..75** | Total Multicast Bytes Received (upper) | | | |
| **76..79** | Total Multicast Bytes Received (lower) | | | |
| **80..83** | Total Broadcast Bytes Received (upper) | | | |
| **84..87** | Total Broadcast Bytes Received (lower) | | | |
| **88..91** | Total Unicast Bytes Transmitted (upper) | | | |
| **92..95** | Total Unicast Bytes Transmitted (lower) | | | |
| **96..99** | Total Multicast Bytes Transmitted (upper) | | | |
| **100..103** | Total Multicast Bytes Transmitted (lower) | | | |
| **104..107** | Total Broadcast Bytes Transmitted (upper) | | | |
| **108..111** | Total Broadcast Bytes Transmitted (lower) | | | |
| **112..115** | Checksum | | | |

### 8.4.78.1 Counter Sizes field

The NC shall indicate in the Counter Sizes field whether the implementation uses 32-bit counters or 64-bit counters in those counter fields above that are defined as 64-bit.

**Table 162 – Counter Sizes field format**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | Total Bytes Received | `0b` = 32-bit<br>`1b` = 64-bit |
| 1 | Total Bytes Transmitted | `0b` = 32-bit<br>`1b` = 64-bit |
| 2 | Total Unicast Bytes Received | `0b` = 32-bit<br>`1b` = 64-bit |
| 3 | Total Multicast Bytes Received | `0b` = 32-bit<br>`1b` = 64-bit |
| 4 | Total Broadcast Bytes Received | `0b` = 32-bit<br>`1b` = 64-bit |
| 5 | Total Unicast Bytes Transmitted | `0b` = 32-bit<br>`1b` = 64-bit |
| 6 | Total Multicast Bytes Transmitted | `0b` = 32-bit<br>`1b` = 64-bit |

| Bit Position | Field Description | Value Description |
|---|---|---|
| 7 | Total Broadcast Bytes Transmitted | `0b` = 32-bit<br>`1b` = 64-bit |

#### 8.4.78.2 Counters Cleared from Last Read field

The NC shall indicate in the Counters Cleared from Last Read field whether the corresponding fields have been cleared since they were last read over NC-SI.

**Table 163 – Counters Cleared from Last Read field format**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | Total Bytes Received | `0b` = Not Cleared<br>`1b` = Cleared |
| 1 | Total Bytes Transmitted | `0b` = Not Cleared<br>`1b` = Cleared |
| 2 | Total Unicast Packets Received | `0b` = Not Cleared<br>`1b` = Cleared |
| 3 | Total Multicast Packets Received | `0b` = Not Cleared<br>`1b` = Cleared |
| 4 | Total Broadcast Packets Received | `0b` = Not Cleared<br>`1b` = Cleared |
| 5 | Total Unicast Packets Transmitted | `0b` = Not Cleared<br>`1b` = Cleared |
| 6 | Total Multicast Packets Transmitted | `0b` = Not Cleared<br>`1b` = Cleared |
| 7 | Total Broadcast Packets Transmitted | `0b` = Not Cleared<br>`1b` = Cleared |
| 8 | Total Unicast Bytes Received | `0b` = Not Cleared<br>`1b` = Cleared |
| 9 | Total Multicast Bytes Received | `0b` = Not Cleared<br>`1b` = Cleared |
| 10 | Total Broadcast Bytes Received | `0b` = Not Cleared<br>`1b` = Cleared |
| 11 | Total Unicast Bytes Transmitted | `0b` = Not Cleared<br>`1b` = Cleared |
| 12 | Total Multicast Bytes Transmitted | `0b` = Not Cleared<br>`1b` = Cleared |
| 13 | Total Broadcast Bytes Transmitted | `0b` = Not Cleared<br>`1b` = Cleared |
| 15..14 | Reserved | |

### 8.4.79 Get Partition Statistics response for FCoE (0xAF)

In the absence of any errors, the channel shall process and respond to the Get Partition Statistics Command and send the response packet shown below when the Stats Type indicates FCoE.

Currently no command-specific reason code is identified for this response.

**Table 164 – Get Partition Statistics (FCoE) response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Stats Type | Counter Sizes | Counters Cleared | |
| **24..27** | Total FCoE Bytes Received (upper) | | | |
| | Total FCoE Bytes Received (lower) | | | |
| | Total FCoE Bytes Transmitted (upper) | | | |
| | Total FCoE Bytes Transmitted (lower) | | | |
| | Total FCoE Packets Received (upper) | | | |
| | Total FCoE Packets Received (lower) | | | |
| | Total FCoE Packets Transmitted (upper) | | | |
| | Total FCoE Packets Transmitted (lower) | | | |
| | Checksum | | | |

#### 8.4.79.1 Counter Sizes field

The NC shall indicate in the Counter Sizes field whether the implementation uses 32-bit counters or 64-bit counters in those counter fields above that are defined as 64-bit.

**Table 165 – Counter Sizes field format**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | Total FCoE Bytes Received | 0b = 32-bit<br>1b = 64-bit |
| 1 | Total FCoE Bytes Transmitted | 0b = 32-bit<br>1b = 64-bit |
| 2 | Total FCoE Packets Received | 0b = 32-bit<br>1b = 64-bit |
| 3 | Total FCoE Packets Received | 0b = 32-bit<br>1b = 64-bit |
| 4..7 | Reserved | Reserved |

#### 8.4.79.2 Counters Cleared from Last Read

The NC shall indicate in the Counters Cleared from Last Read field whether the corresponding fields have been cleared since they were last read over NC-SI.

**Table 166 – Counters Cleared from Last Read field format**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | Total FCoE Bytes Received | `0b` = Not Cleared<br>`1b` = Cleared |
| 1 | Total FCoE Packets Transmitted | `0b` = Not Cleared<br>`1b` = Cleared |
| 2 | Total FCoE Packets Received | `0b` = Not Cleared<br>`1b` = Cleared |
| 3 | Total FCoE Packets Transmitted | `0b` = Not Cleared<br>`1b` = Cleared |
| 15..4 | Reserved | Reserved |

### 8.4.80 Get Partition Statistics response for iSCSI (`0xAF`)

In the absence of any errors, the channel shall process and respond to the Get Partition Statistics Command and send the response packet shown below when the Stats Type indicates iSCSI.

Currently no command-specific reason code is identified for this response.

**Table 167 – Get Partition Statistics (iSCSI) response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Stats Type | Counter Sizes | Counters Cleared | |
| **24..27** | Total iSCSI Offload Bytes Received (upper) | | | |
| | Total iSCSI Offload Bytes Received (lower) | | | |
| | Total iSCSI Offload Bytes Transmitted (upper) | | | |
| | Total iSCSI Offload Bytes Transmitted (lower) | | | |
| | Total iSCSI Offload PDUs Received (upper) | | | |
| | Total iSCSI Offload PDUs Received (lower) | | | |
| | Total iSCSI Offload PDUs Transmitted (upper) | | | |
| | Total iSCSI Offload PDUs Transmitted (lower) | | | |
| | Checksum | | | |

#### 8.4.80.1 Counter Sizes field

The NC shall indicate in the Counter Sizes field whether the implementation uses 32-bit counters or 64-bit counters in those counter fields above that are defined as 64-bit.

**Table 168 – Counter Sizes field format**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | Total iSCSI Offload Bytes Received | `0b` = 32-bit<br>`1b` = 64-bit |
| 1 | Total iSCSI Offload Bytes Transmitted | `0b` = 32-bit<br>`1b` = 64-bit |
| 2 | Total iSCSI Offload PDUs Received | `0b` = 32-bit<br>`1b` = 64-bit |
| 3 | Total iSCSI Offload PDUs Transmitted | `0b` = 32-bit<br>`1b` = 64-bit |
| 4..7 | Reserved | Reserved |

#### 8.4.80.2 Counters Cleared from Last Read

The NC shall indicate in the Counters Cleared from Last Read field whether the corresponding fields have been cleared since they were last read over NC-SI.

**Table 169 – Counters Cleared from Last Read field format**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | Total iSCSI Offload Bytes Received | `0b` = Not Cleared<br>`1b` = Cleared |
| 1 | Total iSCSI Offload Bytes Transmitted | `0b` = Not Cleared<br>`1b` = Cleared |
| 2 | Total iSCSI Offload PDUs Received | `0b` = Not Cleared<br>`1b` = Cleared |
| 3 | Total iSCSI Offload PDUs Transmitted | `0b` = Not Cleared<br>`1b` = Cleared |
| 15..4 | Reserved | Reserved |

### 8.4.81 Get Partition Statistics response for InfiniBand (`0xAF`)

In the absence of any errors, the channel shall process and respond to the Get Partition Statistics Command and send the response packet shown below when the Stats Type indicates InfiniBand.

Currently no command-specific reason code is identified for this response.

**Table 170 – Get Partition Statistics (IB) response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Stats Type | Counter Sizes | Counters Cleared | |
| **24..27** | Total Unicast Packets Received (upper) | | | |
| **28..31** | Total Unicast Packets Received (lower) | | | |
| **32..35** | Total Multicast Packets Received (upper) | | | |
| **36..39** | Total Multicast Packets Received (lower) | | | |
| **40..43** | Total Unicast Packets Transmitted (upper) | | | |
| **44..47** | Total Unicast Packets Transmitted (lower) | | | |
| **48..51** | Total Multicast Packets Transmitted (upper) | | | |
| **52..55** | Total Multicast Packets Transmitted (lower) | | | |
| **56..59** | Total Unicast Bytes Received (upper) | | | |
| **60..63** | Total Unicast Bytes Received (lower) | | | |
| **64..67** | Total Multicast Bytes Received (upper) | | | |
| **68..71** | Total Multicast Bytes Received (lower) | | | |
| **72..75** | Total Unicast Bytes Transmitted (upper) | | | |
| **76..79** | Total Unicast Bytes Transmitted (lower) | | | |
| **80..83** | Total Multicast Bytes Transmitted (upper) | | | |
| **84..87** | Total Multicast Bytes Transmitted (lower) | | | |
| **88..91** | Checksum | | | |

### 8.4.81.1 Counter Sizes field

The NC shall indicate in the Counter Sizes field whether the implementation uses 32-bit counters or 64-bit counters in those counter fields above that are defined as 64-bit.

**Table 171 – Counter Sizes field format**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | Total Unicast Packets Received | 0b = 32-bit<br>1b = 64-bit |
| 1 | Total Unicast Packets Transmitted | 0b = 32-bit<br>1b = 64-bit |
| 2 | Total Multicast Packets Received | 0b = 32-bit<br>1b = 64-bit |
| 3 | Total Multicast Packets Transmitted | 0b = 32-bit<br>1b = 64-bit |

| Bit Position | Field Description | Value Description |
|---|---|---|
| 4 | Total Unicast Bytes Received | 0b = 32-bit<br>1b = 64-bit |
| 5 | Total Unicast Bytes Transmitted | 0b = 32-bit<br>1b = 64-bit |
| 6 | Total Multicast Bytes Received | 0b = 32-bit<br>1b = 64-bit |
| 7 | Total Broadcast Bytes Transmitted | 0b = 32-bit<br>1b = 64-bit |

#### 8.4.81.2 Counters Cleared from Last Read

The NC shall indicate in the Counters Cleared from Last Read field whether the corresponding fields have been cleared since they were last read over NC-SI.

**Table 172 – Counters Cleared from Last Read field format**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | Total Unicast Packets Received | 0b = Not Cleared<br>1b = Cleared |
| 1 | Total Multicast Packets Received | 0b = Not Cleared<br>1b = Cleared |
| 2 | Total Unicast Packets Transmitted | 0b = Not Cleared<br>1b = Cleared |
| 3 | Total Multicast Packets Transmitted | 0b = Not Cleared<br>1b = Cleared |
| 4 | Total Unicast Bytes Received | 0b = Not Cleared<br>1b = Cleared |
| 5 | Total Multicast Bytes Received | 0b = Not Cleared<br>1b = Cleared |
| 6 | Total Unicast Bytes Transmitted | 0b = Not Cleared<br>1b = Cleared |
| 7 | Total Multicast Bytes Transmitted | 0b = Not Cleared<br>1b = Cleared |
| 15..8 | Reserved | |

### 8.4.82 Get Partition Statistics response for RDMA (0xAF)

In the absence of any errors, the channel shall process and respond to the Get Partition Statistics Command and send the response packet shown below when the Stats Type indicates RDMA.

Currently no command-specific reason code is identified for this response.

**Table 173 – Get Partition Statistics (RDMA) response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Stats Type | Counter Sizes | Counters Cleared | |
| **24..27** | Total RDMA Bytes Received (upper) | | | |
| **28..31** | Total RDMA Bytes Received (lower) | | | |
| **32..35** | Total RDMA Bytes Transmitted (upper) | | | |
| **36..39** | Total RDMA Bytes Transmitted (lower) | | | |
| **40..43** | Total RDMA Packets Received (upper) | | | |
| **44..47** | Total RDMA Packets Received (lower) | | | |
| **48..51** | Total RDMA Packets Transmitted (upper) | | | |
| **52..55** | Total RDMA Packets Transmitted (lower) | | | |
| **56..59** | Total Read Request Packets Transmitted (upper) | | | |
| **60..63** | Total Read Request Packets Transmitted (lower) | | | |
| **64..67** | Total Send Packets Transmitted (upper) | | | |
| **68..71** | Total Send Packets Transmitted (lower) | | | |
| **72..75** | Total Write Packets Transmitted (upper) | | | |
| **76..79** | Total Write Packets Transmitted (lower) | | | |
| **80..83** | Checksum | | | |

### 8.4.82.1 Counter Sizes

The NC shall indicate in the Counter Sizes field whether the implementation uses 32-bit counters or 64-bit counters in those counter fields above that are defined as 64-bit.

**Table 174 – Counter Sizes field format**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | Total RDMA Bytes Received | `0b` = 32-bit<br>`1b` = 64-bit |
| 1 | Total RDMA Bytes Transmitted | `0b` = 32-bit<br>`1b` = 64-bit |
| 2 | Total RDMA Packets Received | `0b` = 32-bit<br>`1b` = 64-bit |
| 3 | Total RDMA Packets Transmitted | `0b` = 32-bit<br>`1b` = 64-bit |
| 4 | Total Read Request Packets Transmitted | `0b` = 32-bit<br>`1b` = 64-bit |

| Bit Position | Field Description | Value Description |
|---|---|---|
| 5 | Total Send Packets Transmitted | 0b = 32-bit<br>1b = 64-bit |
| 6 | Total Write Packets Transmitted | 0b = 32-bit<br>1b = 64-bit |
| 7 | Reserved | |

#### 8.4.82.2 Counters Cleared from Last Read

The NC shall indicate in the Counters Cleared from Last Read field whether the corresponding fields have been cleared since they were last read over NC-SI.

**Table 175 – Counters Cleared from Last Read field format**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | Total RDMA Bytes Received | 0b = Not Cleared<br>1b = Cleared |
| 1 | Total RDMA Bytes Transmitted | 0b = Not Cleared<br>1b = Cleared |
| 2 | Total RDMA Packets Received | 0b = Not Cleared<br>1b = Cleared |
| 3 | Total RDMA Packets Transmitted | 0b = Not Cleared<br>1b = Cleared |
| 4 | Total Read Request Packets Transmitted | 0b = Not Cleared<br>1b = Cleared |
| 5 | Total Send Packets Transmitted | 0b = Not Cleared<br>1b = Cleared |
| 6 | Total Write Packets Transmitted | 0b = Not Cleared<br>1b = Cleared |
| 15..7 | Reserved | |

### 8.4.83 Get Partition Statistics Response for Fibre Channel (0xAF)

The channel shall, in the absence of a checksum error or identifier mismatch, always accept the Get Partition Statistics command and send a response when the Stats Type indicates FC.

Table 176 illustrates the packet format of the Get FC Statistics Response. Note that all counters for FC statistics are 32-bit counters.

**Table 176 – Get Partition Statistics (FC) Response packet**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Stats Type | Reserved | Counters Cleared from Last Read | |

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **24..27** | Total FC Frames Received | | | |
| **28..31** | Total FC Frames Transmitted | | | |
| **32..35** | Receive KB Count | | | |
| **36..39** | Transmit KB Count | | | |
| **40..43** | FC Sequences Received | | | |
| **44..47** | FC Sequences Transmitted | | | |
| **48..51** | Link Failures | | | |
| **52..55** | Loss of Signal | | | |
| **56..59** | Invalid CRCs | | | |
| **60..63** | Checksum | | | |

### 8.4.83.1 Counters Cleared from Last Read field

The FC Controller shall also indicate in the Counters Cleared from Last Read field whether the corresponding fields has been cleared since it was last read via NC-SI. The Counters Cleared from Last Read fields should have the format shown in Table 177.

**Table 177 – Counters Cleared from Last Read field format**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | Total FC Frames Received | `0b` = Not Cleared<br>`1b` = Cleared |
| 1 | Total FC Frames Transmitted | `0b` = Not Cleared<br>`1b` = Cleared |
| 2 | Receive KB Count | `0b` = Not Cleared<br>`1b` = Cleared |
| 3 | Transmit KB Count | `0b` = Not Cleared<br>`1b` = Cleared |
| 4 | FC Sequences Received | `0b` = Not Cleared<br>`1b` = Cleared |
| 5 | FC Sequences Transmitted | `0b` = Not Cleared<br>`1b` = Cleared |
| 6 | Link Failures | `0b` = Not Cleared<br>`1b` = Cleared |
| 7 | Loss of Signal | `0b` = Not Cleared<br>`1b` = Cleared |
| 8 | Invalid CRCs | `0b` = Not Cleared<br>`1b` = Cleared |
| 15..9 | Reserved | |

### 8.4.83.2 FC Statistics Counter definitions

**Table 178 – FC Statistics**

| Name | Meaning |
|---|---|
| Total FC Frames Received | Counts the number of FC frames received by the port |
| Total FC Frames Transmitted | Counts the number of FC frames transmitted by the port |
| Receive KB Count | Counts the number of kilobytes transmitted by the port |
| Transmit KB Count | Counts the number of kilobytes transmitted by the port |
| FC Sequences Received | Counts the number of FC sequences received by the port |
| FC Sequences Transmitted | Counts the number of FC sequences transmitted by the port |
| Link Failures | Counts the number of times the link has failed. |
| Loss of Signal | Counts the number of times the signal was lost. |
| Invalid CRCs | Counts the number of CRC errors detected. |

### 8.4.84 Set Module Management Data command (`0x30`)

Set Module Management Data command is used to write management data to modules plugged into the NC. Set Module Management Data is defined as a channel command (that is, the command is sent with a valid Package ID and a valid Internal Channel ID) addressed to a particular module plugged into the port corresponding to the channel.

The writing of management data to the module is implementation dependent.

A two-byte Type identifier is used to specify the bank and page index of the target data to be returned. The previous SFF-type specifications do not use the term ‘bank’, instead they use upper- and lower-page terminology.

For this command, the lower page is considered Bank 0 and the upper page Bank 1. Some devices only support one bank and therefore will only respond with data with the bank index set to 0x00.

Table 179 illustrates the packet format for the Set Module Management Data command.

**Table 179 – Set Module Management Data command packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Requested Bank | Requested Page | Offset | Length |
| **20..N-1** | Management Data | | | |
| **N..N+3** | Checksum | | | |
| | Pad (if N ≤ 38) | | | |

#### 8.4.84.1 Requested Bank field

The Requested Bank field is the value of the bank where the management data is being requested to be written.

#### 8.4.84.2 Requested Page field

The Requested Page field is the value of the page where the management data is being requested to be written.

#### 8.4.84.3 Offset field

This field shall be set to the offset within the targeted bank of the targeted page where the management data is written.

#### 8.4.84.4 Length field

This field shall be set to the length of the management data included in the payload. The Length shall be in bytes and multiple of 4. The length shall be less than or equal to 128.

#### 8.4.84.5 Management Data field

The management data that is requested to be written.

### 8.4.85 Set Module Management Data response (0xB0)

In the absence of any errors, the NC shall process and respond to the Set Module Management Data Command and send the response packet shown in Table 180.

Currently no command-specific reason code is identified for this response.

If there is no module installed or module is not present, then the NC shall return response/reason codes Command Unavailable/Information not available.

The NC shall fail this command and return response and reason codes as Command Unsupported and Unknown/Unsupported Command Type respectively for backplane and RJ-45 implementations.

If the Requested Bank or Page number does not exist, then the NC should return the Command Failed response code and should return reason code Parameter Is Invalid, Unsupported, or Out-of-Range.

If the module is resetting or powering up, then the NC shall return the Command Failed response code and should return reason code Information not available.

If the module is powered down, then the NC shall return the Command Failed response code and should return reason code Secondary Device Not Powered.

If the location at which the management data is being requested to be written is not writable, then the NC should return the Command Failed response code and should return reason code No Reason Code or Parameter Is Invalid, Unsupported, or Out-of-Range.

If the module cannot write the management data in the allocated time, then the NC shall return response/reason code either Command Failed/Command Timeout or Delayed Response/Command Timeout.

It is highly recommended that the MC that plans to use this command enables Delayed Response feature.

**Table 180 – Set Module Management Data response packet format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.4.86 Get FC Link Status command (0x31)

The Get FC Link Status command allows the Management Controller to query the channel for potential link status and error conditions (see Table 181).

Implementation of this command is conditional and is required only for controllers supporting native Fibre Channel.

Implementation Note:

Some controllers may include a port trunking (bonding) capability in which one (or more) channels will map to multiple physical ports. FC trunking (bonding) is based on the following rules:

- FC controllers provide a maximum of 4 physical ports
- All ports are configured to the same speed
- If trunking is enabled, all ports become involved in a bond, no standalone ports remain
- Ports may bond in pairs or all together
- Dual port controllers bond Ports 1&2 and present one channel to the MC
- Quad port controllers bond Ports (1&2) [trunk 1] and {3&4} [trunk2] or {1&2&3&4} and present two or one channel(s) respectively

**Table 181 – Get FC Link Status command packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Reserved | Reserved | Reserved | Reserved |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.4.87 Get FC Link Status Response `(0xB1)`

The channel shall, in the absence of a checksum error or identifier mismatch, always accept the Get FC Link Status command and send a response (see Table 182).

**Table 182 – Get FC Link Status Response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Num FC Ports | FC Trunk Status | FC Link Status | Trunk Speeds |
| **24..27** | Channel 1 Link Speed | Channel 2 Link Speed | Channel 3 Link Speed | Channel 4 Link Speed |
| **28..31** | Checksum | | | |
| **32..45** | Pad | | | |

#### 8.4.87.1 Num FC Ports field

This is an integer value that specifies the total number of physical ports on the Package.

#### 8.4.87.2 FC Trunk Status field

This field indicates if the physical port is a member of a FC trunk.

**Table 183 – FC Trunk Status field bit definitions**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | Port 1 Trunk Flag | `0b` = Physical Port 1 Is not a member of a trunk<br>`1b` = Physical Port 1 Is a member of a trunk |
| 1 | Port 2 Trunk Flag | `0b` = Physical Port 2 Is not a member of a trunk<br>`1b` = Physical Port 2 Is a member of a trunk |
| 2 | Port 3 Trunk Flag | `0b` = Physical Port 3 Is not a member of a trunk<br>`1b` = Physical Port 3 Is a member of a trunk |
| 3 | Port 4 Trunk Flag | `0b` = Physical Port 4 Is not a member of a trunk<br>`1b` = Physical Port 4 Is a member of a trunk |

| Bit Position | Field Description | Value Description |
|---|---|---|
| 7..4 | Reserved | None |

#### 8.4.87.3 FC Link Status field

Table 184 describes the FC Link Status field bit definitions.

**Table 184 – FC Link Status field bit definitions**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | Port 1 Link Flag | 0b = Physical Port 1 Link is down<br>1b = Physical Port 1 Link is up |
| 1 | Port 2 Link Flag | 0b = Physical Port 2 Link is down<br>1b = Physical Port 2 Link is up |
| 2 | Port 3 Link Flag | 0b = Physical Port 3 Link is down<br>1b = Physical Port 3 Link is up |
| 3 | Port 4 Link Flag | 0b = Physical Port 4 Link is down<br>1b = Physical Port 4 Link is up |
| 7..5 | Reserved | None |

#### 8.4.87.4 Trunk Speeds field

The percentage of the configured trunk speed that is currently available represented as an integer.

Table 185 describes the Trunk Speeds field.

**Table 185 – Trunk Speeds field**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 3..0 | Trunk 1 Percentage Speed | Percentage of the Trunk 1 configured link speed that is available expressed as hex value. Not applicable if no Trunks are configured.<br>0x0 = 0%<br>0x1 = 25%<br>0x2 = 50%<br>0x3 = 75%<br>0x4 = 100%<br>0x5-0xf = Reserved |

| Bit Position | Field Description | Value Description |
|---|---|---|
| 7..4 | Trunk 2 Percentage Speed | Percentage of the Trunk 2 configured link speed that is available (expressed as hex value. Not applicable if two Trunks are not configured.-<br>0x0 = 0%<br>0x1 = 25%<br>0x2 = 50%<br>0x3 = 75%<br>0x4 = 100%<br>0x5-0xf = Reserved |

### 8.4.87.5 Channel Link Speed field

The Channel Link Speed field provides a link speed based on NC-SI Channel configuration. Up to 4 Channel link speed fields are supported. If the number of FC ports is equal to the number of reported NC-SI channels, then trunking is not active, and the reported speed is the speed of the channel on the port. In two- or four-port trunking modes, the number of FC ports will be twice or four times the number of reported NC-SI channels and the reported configured link speed is the sum of the individual link speeds in the trunk. If one or more of the member links goes down the reported link speed will not change, but the FC Link Status and Trunk Speed fields will provide the indication that the trunk is not operating at its stated speed.

Table 186 describes the Channel Link Speed field bit definitions.

**Table 186 – Channel Link Speed field**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 3..0 | Link Speed | 0x0 = No link speed established<br>0x1 = FC2<br>0x2 = FC4<br>0x3 = FC8<br>0x4 = FC16<br>0x5 = FC32<br>0x6 = FC64<br>0x7 = FC128<br>0x8 = FC256<br>0x9-0xf = Reserved |
| 7..4 | Reserved | None |

### 8.4.88 Get Module Management Data command (0x32)

The Get Module Management Data command is used to retrieve 128-byte blocks of management and inventory data stored in the passive copper cable or optical transceiver module associated with the channel. Different standards and specifications exist (e.g., +SFF and CMIS) in the industry for this management data, but they share common data access methods allowing this command to successfully operate with the known variety of module interface specifications.

A two-byte Type identifier is used to specify the bank and page index of the target data to be returned. The older SFF-type specifications do not use the term 'bank', instead they use upper and lower page terminology. For this command the lower page is considered Bank 0 and the upper page Bank 1. Some devices only support 1 bank and therefore will only respond with data with the bank index set to 0x00.

The lower 128 bytes of page 00h typically contains more important time-critical data. The upper 128 bytes of page 00h contains static inventory information. The implementation may read and cache the upper 128 bytes once upon power on or module insertion to expedite processing of requests for page 00h data.

For a given module, the NC shall support reading of all mandatory pages defined by the transceiver's Management Data specification. The reading of optional and Vendor-defined pages and any writing of pages is implementation dependent.

Table 136 illustrates the packet format for the Get Module Management Data command.

**Table 187 – Get Module Management Data command packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Requested Bank | Requested Page | Reserved | Flags |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

#### 8.4.88.1 Requested Bank field

The Requested Bank field is the value of the bank data being requested.

#### 8.4.88.2 Requested Page field

The Requested Page field is the value of the page data being requested.

#### 8.4.88.3 Flags field

**Table 188 – Flag field bit definitions**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 0 | Page Upper Flag | 0b = Requesting lower page data<br>1b = Requesting upper page data |
| 7..1 | Reserved | None |

### 8.4.89 Get Module Management Data response (`0xB2`)

In the absence of any errors, the NC shall process and respond to the Get Module Management Data Command and send the response packet shown in Table 137.

Currently no command-specific reason code is identified for this response.

If there is no module installed or module is not present, then the NC shall return response/reason codes Command Unavailable/Information not available.

If the Requested Bank or Page number does not exist, then the NC should return the Command Failed response code with the reason code Parameter Out-of-Range.

The NC shall return the Command Failed response code with the following reason codes for the conditions below:

- 
- If the module is resetting or powering up, then the NC shall return reason code Information not available.
- If the module is powered down, then the NC shall return reason code Secondary Device Not Powered.

If the module cannot respond with data in the allocated time, then the NC shall either return Command Timeout or Delayed Response as supported by the implementation.

The NC shall fail this command and return response and reason codes as Command Failed and Information not available respectively for backplane and RJ-45 implementations.

**Table 189 – Get Module Management Data response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Max Bank | Max Page | Bank Number | Page Number |
| **24..27** | Data$_0$ | Data$_1$ | … | |
| **48..51** | Checksum | | | |

#### 8.4.89.1 Max Bank field

The Max Bank field contains the value of the highest Bank number supported for the requested page of the module. If the module type does not support Banks, the field shall be set to `0x00`.

#### 8.4.89.2 Max Page field

The Max Page field contains the value of the highest Page number in the current Bank supported by the module. If the NC has not or cannot determine the highest Page number, then the value of `0xFF` shall be returned.

#### 8.4.89.3 Bank Number field

The Bank Number field contains the value of the Bank number requested by the command.

#### 8.4.89.4 Page Number field

The Page Number field contains the value of the Page number requested by the command.

#### 8.4.89.5 Module Type Decode

SFF-8024 provides a mapping of module types, their identifiers reported in codes and the Management Interface Specification they comply with.

**Table 190 – Module Type definitions**

| Identifier | Form Factor | Management Interface Specification |
|---|---|---|
| 0x02 | Module soldered to PCB | SFF-8472 |
| 0x03 | SFP/SFP+ / SFP28 and later | SFF-8472 |
| 0x0D | QSFP+ | SFF-8436 |
| 0x11 | QSFP+ / QSFP28 and later | SFF-8636 or CMIS |
| 0x18 | QSFP-DD / QSFP-DD800 | CMIS |
| 0x1E | QSFP+ or later | CMIS |
| 0x19 | OSFP | CMIS |
| 0x1A | SFP-DD | SFP-DD Management Interface Specification |
| 0x1B | DSFP | DSFP |
| 0x17 | MicroQSFP | SFF-8436 |
|  | Reserved | Reserved |

### 8.4.90 Set Pass-through Mode Control Command (0x33)

The Set Pass-through Mode Control command allows the Management controller to enable and disable specified data paths for Pass-through data on the channel when supported by the NC.

Implementation of this command is conditional depending on the type of device and its feature set. For non-Ethernet devices, this command would only be implemented if some type of Pass-thru is supported. For Ethernet NCs, support of either Host-BMC Pass-through or embedded CPU-BMC Pass-through functionality mandates the implementation of this command. Network-BMC Pass-through is traditional NC-SI Pass-through (required in NC-SI), whereas Host-BMC Pass-through is defined to be a network path between the Host and the BMC via the NC-SI Interface. Embedded CPU-BMC Pass-through is defined as a network path that is defined between the BMC and a compute engine or other entity on the network adapter. Further definition of these interfaces is beyond the scope of this specification.

The Host-BMC Pass-through, Network-BMC Pass-through and embedded CPU-BMC Pass-through controls specified in this command act as masks in conjunction with the existing Enable Channel and Enable Channel TX commands. The existing Pass-through MAC address and filtering control methods are simply extended to all defined data paths when configured. No additional filters or MACs are provided.

Table 191 illustrates the packet format for the Set Pass-through Mode Control Command.

**Table 191 – Set Pass-through Mode Control Command**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Reserved | Reserved | Pass-through Type | Reserved |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.4.90.1 Pass-through Type Field

The Pass-through Type field indicates which Pass-through data path is to be enabled or disabled as described in Table 192.

**Table 192 – Pass-through Type definitions**

| Bit | Field Description | Value Description |
|---|---|---|
| 0 | Network-BMC Pass-through traffic | `0b` = Disallowed<br>`1b` = Allowed (default) |
| 1 | Host-BMC Pass-through traffic | `0b` = Disallowed (default)<br>`1b` = Allowed |
| 2 | Embedded CPU-BMC Pass-through traffic | `0b` = Disallowed (default)<br>`1b` = Allowed |
| 7..3 | Reserved | 0b |

### 8.4.91 Set Pass-through Mode Control Response (`0xB3`)

In the absence of any errors, the channel shall process and respond to the Set Pass-through Mode Control command and send the response packet shown in Table 193 – Set Pass-through Mode Control Response Packet.

**Table 193 – Set Pass-through Mode Control Response Packet**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **28..31** | Checksum | | | |
| **32..45** | Pad | | | |

### 8.4.92 Get Pass-through Mode Command (0x34)

The Get Pass-through Mode command allows the Management controller to query the Network Controller for the current state of the Pass-through data paths supported by the channel. Implementation of this command is required if the Set Pass-through Mode Control command is implemented.

Table 194 illustrates the packet format for the Get Pass-through Mode Control command.

**Table 194 – Get Pass-through Mode Command Packet**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

## 8.4.93 Get Pass-through Mode Response (0xB4)

In the absence of any errors, the channel shall process and respond to the Get Pass-through Mode Control command and send the response packet shown in Table 195.

**Table 195 – Get Pass-through Mode Response Packet**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Reserved | Reserved | Pass-through Mode Status | Pass-through Mode Capability |
| **24..27** | Checksum | | | |
| **28..45** | Pad | | | |

### 8.4.93.1 Pass-through Mode Status Field

The Pass-through Mode Status field indicates which Pass-through data path(s) are currently allowed.

**Table 196 – Pass-through Type definitions**

| Bit | Field Description | Value Description |
|---|---|---|
| 0 | Network-BMC Pass-through traffic | 0b = Currently Disallowed<br>1b = Currently Allowed |
| 1 | Host-BMC Pass-through traffic | 0b = Currently Disallowed<br>1b = Currently Allowed |
| 2 | Embedded CPU -BMC Pass-through traffic | 0b = Currently Disallowed<br>1b = Currently Allowed |
| 7..3 | Reserved | 0b |

### 8.4.93.2 Pass-through Mode Capability Field

The Pass-through Mode Capability field indicates which Pass-through Mode data path(s) are supported by the implementation.

**Table 197 – Pass-through Type definitions**

| Bit | Field Description | Value Description |
|---|---|---|
| 0 | Network-BMC Pass-through traffic | `0b` = Not Supported<br>`1b` = Supported |
| 1 | Host-BMC Pass-through traffic | `0b` = Not Supported<br>`1b` = Supported |
| 2 | Embedded CPU-BMC Pass-through traffic | `0b` = Not Supported<br>`1b` = Supported |
| 7..3 | Reserved | 0b |

### 8.4.94 Get VF Allocation command `(0x35)`

The Get VF Allocation command is a Package command that allows the Management controller to receive the current number of PCIe Virtual Functions being advertised by each Physical Function in PCIe Configuration Space.,

See the Set VF Allocation command description for additional information.

Table 198 illustrates the packet format of the Get VF Allocation Command.

**Table 198 – Get VF Allocation Command Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

### 8.4.95 Get VF Allocation Response `(0xB5)`

In the absence of any errors, the package shall process and respond to the Get VF Allocation command and send the response packet shown in the table below.

**Table 199 – Get VF Allocation Response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Function 0 Num VFs | Function 1 Num VFs | Function 2 Num VFs | Function 3 Num VFs |
| **24..27** | Function 4 Num VFs | … | | |
| | … | | | |
| | Checksum | | | |
| | Pad | | | |

#### 8.4.95.1 Function Num VFs field

Field entries contain the number of VFs that each Physical Function is advertising in Configuration Space.

**Table 200 – Function Num VFs Fields**

| | Field Description | Value Description |
|---|---|---|
| | Function 0 Num VFs | Number of VFs currently being advertised by Function 0 |
| | Function 1 Num VFs | Number of VFs currently being advertised by Function 1 |
| | … | … |

### 8.4.96 Set VF Allocation command `(0x36)`

The Set VF Allocation command is a Package command that allows the Management controller to configure the number of PCIe Virtual Functions to be advertised in PCIe Configuration Space by each of the Physical Functions in the NC. The total number of Virtual Functions the NC supports is returned in the Get NC Capabilities and Settings response and the sum of the VFs configured by this command shall not exceed that total value.

The values configured by this command are held by the controller and only take effect at the next PCIe reset. The configuration is persistent unless changed by another Set VF Allocation command or other mechanism.

Table 201 illustrates the packet format of the Set VF Allocation Command.

**Table 201 – Set VF Allocation Command packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Function 0 Num VFs | Function 1 Num VFs | Function 2 Num VFs | Function 3 Num VFs |
| **…** | Function 4 Num VFs | … | | |
| | … | | | |
| | Checksum | | | |
| | Pad | | | |

### 8.4.96.1 Function Num VFs field

Field entries contain the number of VFs that each Physical Function is advertising in Configuration Space

**Table 202 – Function Num VFs Fields**

| | Field Description | Value Description |
|---|---|---|
| | Function 0 Num VFs | Number of VFs to be advertised by Function 0 |
| | Function 1 Num VFs | Number of VFs to be advertised by Function 1 |
| … | … | … |
| | Function N Num VFs | Number of VFs to be advertised by Function N |

## 8.4.97 Set VF Allocation Response (0xB6)

In the absence of any errors, the channel shall process and respond to the Set VF Allocation Command and send the response packet shown in Table 203.

**Table 203 – Set VF Allocation Response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **24..27** | Checksum | | | |
| **36..39** | Pad | | | |

## 8.4.98 Get InfiniBand Link Status command (0x38)

The Get InfiniBand Link Status command allows the Management Controller to query the channel for the IB link status. In addition to the generic Get Link Status command, this command provides IB specific link status.

Implementation of this command is conditional and is required only for controllers supporting InfiniBand.

Table 204 illustrates the packet format of the InfiniBand Link Status command.

**Table 204 – Get InfiniBand Link Status command**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

### 8.4.99 Get InfiniBand Link Status Response (0xB8)

The channel shall, in the absence of a checksum error or identifier mismatch, always accept the Get InfiniBand Link Status command and send a response.

The Get InfiniBand Link Status Response frame reports IB link width, logical and physical link states, and the supported and the configured link speed of the port.

Table 205 illustrates the packet format of the Get InfiniBand Link Status Response.

**Table 205 – Get InfiniBand Link Status Response packet**

<table>
<tr><th></th><th colspan="5">Bits</th></tr>
<tr><th>Bytes</th><th>31..24</th><th>23..16</th><th>15..08</th><th colspan="2">07..00</th></tr>
<tr><td><b>00..15</b></td><td colspan="5">NC-SI Header</td></tr>
<tr><td><b>16..19</b></td><td colspan="2">Response Code</td><td colspan="3">Reason Code</td></tr>
<tr><td><b>28..31</b></td><td>IB Link Active Width</td><td>IB Link Supported Width</td><td>Link Type</td><td>Phys State</td><td>Logical Port State</td></tr>
<tr><td><b>32..35</b></td><td>Reserved</td><td>IB Link Active Speed</td><td>Reserved</td><td colspan="2">IB Link Supported Speed</td></tr>
<tr><td><b>36..39</b></td><td colspan="5">Checksum</td></tr>
<tr><td><b>40..45</b></td><td colspan="5">Pad</td></tr>
</table>

**Table 206 – InfiniBand Link Status definitions**

| Name | Direction | Description |
|---|---|---|
| IB Link Active Width | TX | When Link Type is InfiniBand and physical link is up, this field reflects the active link width. Otherwise, this field returns 0b.<br>Bit 0 – `1b` = 1X link width<br>Bit 1 – `1b` = 2X link width<br>Bit 2 – `1b` = 4X link width<br>Bit 3 – `1b` = 8X link width<br>Bits 7:4 Reserved |

| Name | Direction | Description |
|---|---|---|
| IB Link Supported Width | RX | When Link Type is InfiniBand, this field reflects the supported link widths. When Link Type is Ethernet, this field returns 0.<br>Bit 0 – 1b = 1X link width is supported<br>Bit 1 – 1b = 2X link width is supported<br>Bit 2 – 1b = 4X link width is supported<br>Bit 3 – 1b = 8X link width is supported<br>Bits 7:4 Reserved |
| Link Type | TX | Reflects the configured link type.<br>Bit 0 – 0b = Ethernet<br>1b = InfiniBand<br>Bits 7:1 Reserved |
| Phys State | RX | The physical link state as specified in IB spec (PortInfoPortPhysicalState)<br>0x0 = Used when Link Type is Ethernet<br>0x1 = Sleep<br>0x2 = Polling<br>0x3 = Disabled<br>0x4 = PortConfigurationTraining<br>0x5 = LinkUp<br>0x6 = LinkErrorRecovery<br>0x7 = PhyTest |
| Logical Port State | TX | The logical port state of the physical port as specified in IB spec (PortInfo.PortState)<br>0x0: Used when Link Type is Ethernet<br>0x1: Down<br>0x2: Init<br>0x3: Arm<br>0x4: Active |

| Name | Direction | Description |
|---|---|---|
| IB Link Active Speed | TX | When Link Type is InfiniBand and the physical link is up, this field reflects the active link speed. Otherwise this field returns `0x00`.<br>Bit 0 – `1b` = SDR<br>Bit 1 – `1b` = DDR<br>Bit 2 – `1b` = QDR<br>Bit 3 – `1b` = FDR10<br>Bit 4 – `1b` = FDR<br>Bit 5 – `1b` = EDR<br>Bit 6 – `1b` = HDR<br>Bit 7 – `1b` = NDR |
| IB Link Supported Speed | RX | When Link Type is InfiniBand, this field reflects the supported link speeds. When Link Type is Ethernet this field returns `0x00`.<br>Bit 0 – `1b` = SDR<br>Bit 1 – `1b` = DDR<br>Bit 2 – `1b` = QDR<br>Bit 3 – `1b` = FDR10<br>Bit 4 – `1b` = FDR<br>Bit 5 – `1b` = EDR<br>Bit 6 – `1b` = HDR<br>Bit 7 – `1b` = NDR |

### 8.4.100 Get InfiniBand Statistics command `(0x39)`

The Get IB Statistics command allows the Management Controller to query the channel for the IB Statistics.

Implementation of this command is conditional and is required only for controllers supporting InfiniBand.

Table 207 illustrates the packet format of the Get IB Statistics Command.

**Table 207 – Get InfiniBand Statistics Command**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

### 8.4.101 Get InfiniBand Statistics Response (0xB9)

The channel shall, in the absence of a checksum error or identifier mismatch, always accept the Get IB Statistics command and send a response.

The Get IB Statistics Response frame reports a set of IB statistics from the channel. A value of `0xFFFFFFFF` shall be used for any unsupported counter.

All counters shall be reset on Controller resets or power-cycles only.

Table 208 illustrates the packet format of the Get IB Statistics Response.

**Table 208 – Get InfiniBand Statistics Response packet**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | PortXmitData | | | |
| **24..27** | PortRcvData | | | |
| **28..31** | PortXmitPkts | | | |
| **32..35** | PortRcvPkts | | | |
| **36..39** | PortXmitWait | | | |
| **40..43** | PortXmitDiscard | | | |
| **44..47** | SymbolErrorCounter | | | |
| **48..51** | LinkErrorRecoveryCounter | | | |
| **52..55** | LinkDownedCounter | | | |
| **56..59** | PortRcvErrors | | | |
| **60..63** | PortRcvRemotePhysicalErrors | | | |
| **64..67** | PortRcvSwitchRelayErrors | | | |
| **68..71** | LocalLinkIntegrityErrors | | | |
| **72..75** | ExcessiveBufferOverrun | | | |
| **76..79** | VL15Dropped | | | |
| **80..83** | Checksum (3..2) | | | |

**Table 209 – InfiniBand Statistics Counter definitions**

| Name | Direction | Description |
|---|---|---|
| PortXmitData | TX | Total number of data octets, divided by 4 (lanes), transmitted on all VLs. |
| PortRcvData | RX | Total number of data octets, divided by 4 (lanes), received on all VLs. |
| PortXmitPkts | TX | Total number of packets transmitted on all VLs from this port. This may include packets with errors. |
| PortRcvPkts | RX | Total number of packets (this may include packets containing Errors. |

| Name | Direction | Description |
|---|---|---|
| PortXmitWait | TX | Number of ticks during which the port had data to transmit but no data was sent during the entire tick (either because of insufficient credits or because of lack of arbitration). |
| PortXmitDiscard | TX | Total number of outbound packets discarded by the port because the port is down or congested. |
| SymbolErrorCounter | RX | Total number of minor link errors detected on one or more physical lanes. |
| LinkErrorRecoveryCounter | RX | Total number of times the Port Training state machine has successfully completed the link error recovery process. |
| LinkDownedCounter | RX | Total number of times the Port Training state machine has failed the link error recovery process and downed the link. |
| PortRcvErrors | RX | Total number of packets containing an error that were received on the port. |
| PortRcvRemotePhysicalErrors | RX | Total number of packets marked with the EBP delimiter received on the port. |
| PortRcvSwitchRelayErrors | RX | Total number of packets received on the port that were discarded because they could not be forwarded by the switch relay. |
| LocalLinkIntegrityErrors | RX | Number of times that the count of local physical errors exceeded the threshold specified by LocalPhyErrors. |
| ExcessiveBufferOverrun | RX | Number of times that OverrunErrors consecutive flow control update periods occurred, each having at least one overrun error. |
| VL15Dropped | RX | Number of incoming VL15 packets dropped due to resource limitations (e.g., lack of buffers) of the port. |

### 8.4.102 Settings Commit command (0x47)

The Settings Commit command is a package command used by the Management Controller to indicate that those previously programmed settings defined as persistent must now be written to non-volatile storage. It also indicates that any previously programmed individual settings that have dependencies on other settings (e.g., partition bandwidth) have been fully programmed and can be finalized and/or validated. Only those settings in commands that returned successful response/reason codes will be written to non-volatile storage.

The MC can only be assured that the settings have been persisted when this commit command has a successful completion. It is highly likely that execution of this command will result in a Delayed Response. The MC should assume that all the settings that were sent but not committed are lost on losses of power, various types of resets as defined by the NC, return to initial states of any affected channel, etc. and must be resent after the interruption. The MC is ultimately responsible for ensuring its configuration settings have been properly received by the NC, therefore it is recommended that the MC monitor settings as appropriate.

Table 210 illustrates the packet format of the Settings Commit command.

**Table 210 – Settings Commit command packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

## 8.4.103 Settings Commit response (0xC7)

The package shall, in the absence of an error, always accept the Settings Commit command and send the response packet shown in Table 211.

Currently no command-specific reason code is identified for this response.

**Table 211 – Settings Commit response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

## 8.4.104 Get ASIC Temperature (0x48)

The Get ASIC Temperature command allows the Management controller to query for temperature values from the Controller's on-chip thermal sensor(s).

The Get ASIC Temperature command is defined as a package level command. This means the command shall be addressed to the package (that is, the command is sent with the Internal Channel ID set to `0x1F`).

The internal temperature of the controller is returned in the response of this command. If the controller has multiple internal temperature sensors, the highest measured temperature with respect to its threshold shall be returned.

Table 212 illustrates the packet format of the Get ASIC Temperature Command.

**Table 212 – Get ASIC Temperature Command packet**

| | Bits | | | |
|---|---|---|---|---|
| Bytes | 31..24 | 23..16 | 15..08 | 07..00 |
| 00..15 | NC-SI Header | | | |
| 16..19 | Reserved | | | |
| 20..23 | Checksum | | | |
| 24..45 | Pad | | | |

## 8.4.105 Get ASIC Temperature Response `(0xC8)`

The package shall, in the absence of a checksum error or identifier mismatch, always accept the Get ASIC Temperature Command and send a response.

Table 213 illustrates the packet format of the Get ASIC Temperature Response.

**Table 213 – Get ASIC Temperature Response packet**

| | Bits | | | |
|---|---|---|---|---|
| Bytes | 31..24 | 23..16 | 15..08 | 07..00 |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |
| 20..23 | Maximum temperature | | Current temperature | |
| 24..27 | Checksum | | | |
| 28..45 | Pad | | | |

### 8.4.105.1 Maximum Temperature Value

This value is the maximum T-Diode temperature limit in degrees Celsius at which the controller can operate at full load for its rated service lifetime. The value should be derated to take measurement tolerance into account. The value shall be reported as a signed 16-bit integer.

### 8.4.105.2 Current Temperature Value

This value is the highest current real-time temperature of the ASIC sensors in degrees Celsius. The value shall be reported as a signed 16-bit integer.

## 8.4.106 Get Ambient Temperature `(0x49)`

The Get Ambient Temperature command allows the Management controller to query for temperature values from ambient temperature sensor(s) attached to the Controller.

The Get Ambient Temperature command is defined as a package command.

Table 214 illustrates the packet format of the Get Ambient Temperature command.

**Table 214 – Get Ambient Temperature command packet**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Reserved | | | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

## 8.4.107 Get Ambient Temperature Response (0xC9)

The Package shall, in the absence of a checksum error or identifier mismatch, always accept the Get Ambient Temperature Command and send a response.

Table 215 illustrates the packet format of the Get Ambient Temperature Response.

**Table 215 – Get Ambient Temperature Response packet**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Temperature3 value | Temperature2 Value | Temperature1 Value | Number of sensors |
| **24..27** | Checksum | | | |
| **28..45** | Pad | | | |

### 8.4.107.1 Temperature1, Temperature2, Temperature3 Values

Each temperature value (up to 3 values as specified by the Number of sensors field) is the real time ambient temperature reported in degrees Celsius. If Number of sensors is set to 0, then all three temperature values are invalid and shall be ignored. If the Number of sensors is set to 1, then Temperature 2 and Temperature 3 are invalid and shall be ignored. If Number of sensors is set to 2, then Temperature 3 is invalid and shall be ignored. If Number of sensors is set to 3 or more, then Temperature1, Temperature2, and Temperature3 are valid. Each valid temperature value shall be reported as a signed 8-bit integer. It is possible that the ambient temperature value may exceed the bounds of the 8-bit signed integer. When the value exceeds either the upper or lower bound that can be represented by the signed 8-bit integer, then the value reported shall be one of the matching bound.

## 8.4.108 Get Transceiver Temperature (0x4A)

The Get Transceiver Temperature command allows the Management controller to query for the real time temperature value and thresholds of the (optical) transceiver attached to the channel.

Table 216 illustrates the packet format of the Get Transceiver Temperature Command.

**Table 216 – Get Transceiver Temperature Command Packet**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Reserved | | | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.4.109 Get Transceiver Temperature Response (0xCA)

The channel shall, in the absence of a checksum error or identifier mismatch, always accept the Get Transceiver Temperature command and send a response.

The Get Transceiver Temperature Response frame contains the current temperature of the attached module and the high side temperature thresholds.

Definitions and interpretation of the data fields in the response are defined in the relevant SFF or MSA specification (e.g., SFF-8472, SFF-8436, SFF-8636, CMIS 4.0, 5.x, etc.) for the transceiver. 16-bit values are encoded as one contiguous entity with the most significant bit in bit 15 (or 31) and least significant bit in bit 0 (or 16) in the response packet. The Controller is not expected to modify the data read from the transceiver.

In cases where the transceiver supports more than one channel, each channel shall provide a response when queried.

The reason code Information not available shall be used if the transceiver is not present, does not provide temperature data or if the command is issued before the transceiver has not yet achieved power up state.

Table 217 illustrates the packet format of the Get Transceiver Temperature Response.

**Table 217 – Get Transceiver Temperature Response packet**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Temp High Alarm Threshold | | Temp High Warning Threshold | |
| **24..27** | Temperature Value | | Reserved | |
| **28..31** | Checksum | | | |

### 8.4.110 Thermal Shutdown Control Command (0x4B)

The Thermal Shutdown Control command allows the Management controller to query for the state of or alternatively set or reset the enablement state of the NC's thermal self-shutdown feature. Thermal shutdown is used for damage avoidance when the NC temperature becomes too high. NCs shall indicate the implementation state of this feature in the Get Capabilities response (0x96) Capabilties Flag field bit 7 and implement this command/response only when the feature is present.

The Thermal Shutdown Control command is defined as a package-level command and is sent with the Internal Channel ID set to 0x1F.

Table 218 illustrates the packet format of the Thermal Shutdown Control Command.

Table 218 – Thermal Shutdown Control Command packet

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Reserved | | | Operation |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.4.110.1 Operation Field

The value specified in this field defines the operation required for the NC's shutdown feature. The NC shall support the query operation. The Enable/Disable operations are optional.

Table 219 – Operation field definitions

| Value | Description | Value Description |
|---|---|---|
| 0 | Disable | Thermal self-shutdown shall be disabled on the device |
| 1 | Enable | Thermal self-shutdown shall be enabled on the device |
| 2 | Query | The currently configured shutdown setting shall be returned |
| others | Reserved | None |

## 8.4.111 Thermal Shutdown Control Response (0xCB)

The package shall, in the absence of a checksum error or identifier mismatch, always accept the Thermal Shutdown Control Command and send a response.

The Operating State status provided in the response shall be confirming the state after the execution of the command. If the Config Control state is set to Read-only, any command to enable or disable the feature shall fail with the Parameter Is Invalid reason code. The other fields shall be included in the response with their current setting.

Table 220 illustrates the packet format of the Thermal Shutdown Control Response.

**Table 220 – Thermal Shutdown Control Response packet**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Reserved | Reserved | Status | Shutdown temperature |
| **24..27** | Checksum | | | |
| **28..45** | Pad | | | |

### 8.4.111.1 Shutdown Temperature Value

This value is the unsigned integer temperature value in degrees Celsius at which the NC will shut itself down when reached.

### 8.4.111.2 Status Field

The value returned in this field is the enablement status of the shutdown feature.

**Table 221 – Status field bit definitions**

| Bit | Description | Value Description |
|---|---|---|
| 0 | Operating State | 0b = Thermal self-shutdown is disabled on the device<br>1b = Thermal self-shutdown is enabled on the device |
| 1 | Enable/Disable Support | 0b = Enable/Disable operations for thermal shutdown are not supported<br>1b = Enable/Disable operations for thermal shutdown are supported |
| Others | Reserved | None |

## 8.4.112 Transmit Data to NC command (0x4C)

The Transmit Data to NC command is a package command that allows the MC to transfer an opaque block of data to the NC. The transfer can be initiated by the MC itself or in response to the reception of the Request Data Transfer AEN. In the latter case, the Total Length of Transfer and Data Handle fields (if provided) should be populated from the AEN fields. If the requested Data Handle is not supported, then the Abort opcode shall be used. Blocks of data that exceed the data space available in one NC-SI frame will be broken down into multiple transfers that comply with NC-SI RBT frame size. When multiple transfers are used:

- Transmission ordering shall be maintained.
- All chunks shall be an integer multiple of 32 bits, (i.e., double-word aligned), except for the last which may include padding to make it double-word aligned.
- If the NC detects a transfer error it may request a retransmission of the active chunk, but no other chunks.
- Any processing of the block of data will start only after the successful reception of all transmitted chunks.

The MC and the NC both have the ability to abort the transfer at any time during the transfer by use of the proper opcode or reason code respectively. If the NC loses transfer context due to being reset or other event, or if it detects an out of order chunk number being specified in the command, it shall abort the transfer. Any data transfer that is aborted is deemed to have failed and cannot be resumed. The MC may attempt to repeat the transfer as a new transfer sequence.

Only one active transfer sequence (transmit or receive) is supported at a given time.

Table 222 illustrates the packet format of the Transmit Data to NC command.

**Table 222 – Transmit Data to NC command packet format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Reserved | | | Opcode |
| **20..23** | Offset | | | |
| **24..27** | Data Handle/Chunk Number | | | |
| | Chunk or Part of Data | | | |
| | Checksum | | | |
| | Pad | | | |

#### 8.4.112.1 Opcode field

**Table 223 – Opcode field format**

| Value | Description | Value Description |
|---|---|---|
| 0x1 | Initial Chunk | First block of data in the transfer |
| 0x2 | Middle Chunk | Intermediate block of data in the transfer |
| 0x4 | Final Chunk | Last block of data in the transfer |
| 0x5 | Initial and Final Chunk | First and last block of data in the transfer |
| 0x8 | Abort Transfer | Terminate the transfer |
| Others | Reserved | |

#### 8.4.112.2 Offset

Offset of the current transfer within the larger data block.

#### 8.4.112.3 Data Handle/Chunk number

For the first chunk being transferred (Initial Chunk Opcode), this is an identifier (Data Handle) of the block of data being transferred. For subsequent chunk transfers it is a sequentially incrementing count for the chunk being transferred (equal to 2 for the second chunk transfer, 3 for the third, etc.).

### 8.4.113 Transmit Data to NC response (0xCC)

The package shall, in the absence of a checksum error or identifier mismatch, always accept the Transmit Data to NC command and send a response.

Table 224 illustrates the packet format of the Transmit Data to NC command response.

There are command-specific reason codes identified for this response (see Table 225).

**Table 224 – Transmit Data to NC response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

**Table 225 – Transmit Data to NC command-specific reason codes**

| Value | Description | Comment |
|---|---|---|
| `0x4C01` | Abort Transfer | Returned when the NC is terminating the transfer for unspecified reason |
| `0x4C02` | Invalid Data Handle | Specified Data Handle is invalid or not supported |
| `0x4C03` | Sequence count error | The Chunk Number received is not consecutive with the previous number received. Also results in an aborted transfer. |
| `0x4C04` | Insufficient Storage | NC cannot process or store a data |

## 8.4.114 Receive Data from NC command (`0x4D`)

The Receive Data from NC command is a package command that allows the MC to receive an opaque block of data from the NC. Blocks of data that exceed the data space available in one NC-SI frame will be broken down into multiple transfers that comply with NC-SI RBT frame size. When multiple transfers are used:

- Reception ordering shall be maintained.
- All chunks shall be an integer multiple of 32 bits, (i.e., double-word aligned), except for the last which may include padding to make it double-word aligned.
- If the MC detects a transfer error it may request a retransmission of the active chunk, but no other.
- Any processing of the block of data will start only after the successful reception of all transmitted chunks.

The MC and the NC both have the ability to abort the transfer at any time during the transfer by use of the proper opcode or reason code respectively. If the NC loses transfer context due to being reset or other event, or if it detects an out of order chunk number being specified in the command, it shall abort the transfer. Any data transfer that is aborted is deemed to have failed and cannot be resumed. The MC may attempt to repeat the transfer as a new transfer sequence.

Only one active transfer sequence (transmit or receive) should be supported at a given time.

Table 226 illustrates the packet format of the Receive Data from NC command.

**Table 226 – Receive Data from NC command packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Reserved | | | Opcode |
| **20..23** | Offset | | | |
| **24.27** | Data Handle/Chunk Number | | | |
| **28..31** | Checksum | | | |
| **32..45** | Pad | | | |

### 8.4.114.1 Opcode field

**Table 227 – Opcode field format**

| Value | Description | Value Description |
|---|---|---|
| 0 | Initial Chunk | Request for the first chunk of the transfer to be returned |
| 1 | Reserved | |
| 2 | Next Chunk | Request for the next chunk of the transfer to be returned |
| 3 | Abort Transfer | Termination of transfer by MC |
| Others | Reserved | |

### 8.4.114.2 Offset field

Offset of the current transfer within the larger data block. For a given data transfer, the value of this field shall be the same for all NC-SI commands of the data transfer.

### 8.4.114.3 Data Handle/Chunk number field

For the first chunk being requested (Initial Chunk Opcode), this is an identifier (Data Handle) of the block of data being requested. For subsequent chunk transfers it is a sequentially incrementing count for the chunk being transferred (equal to 2 for the second chunk transfer, 3 for the third, etc.).

**Table 228: Data Handle Values**

| Value | Description | Comment |
|---|---|---|
| `0x00000000-0xFFFEFFFF` | Vendor Defined | Implementation specific |
| `0xFFFF0000` | Core dump | Data Handle used to retrieve core dump |
| `0xFFFF0001` | Crash dump | Data Handle used to retrieve crash dump |
| `0xFFFF0002-0xFFFFFFFF` | DMTF Reserved | Reserved for the future use by the DMTF |

### 8.4.115 Receive Data from NC response (0xCD)

The package shall, in the absence of a checksum error or identifier mismatch, always accept the Receive Data from NC command and send a response.

Table 229 illustrates the packet format of the Receive Data from NC command response.

**Table 229 – Receive Data from NC response packet format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Reserved | | | Opcode |
| | Data | | | |
| | Checksum | | | |
| | Pad (if needed) | | | |

#### 8.4.115.1 Opcode field

**Table 230 – Opcode field format**

| Value | Description | Value Description |
|---|---|---|
| 0x1 | Initial Chunk | First block of data in the transfer |
| 0x2 | Middle Chunk | Intermediate block of data in the transfer |
| 0x4 | Final Chunk | Last block of data in the transfer |
| 0x5 | Initial and Final Chunk | First and last block of data in the transfer |
| 0x8 | Abort Transfer | Terminate the transfer |
| Others | Reserved | |

**Table 231 – Receive Data from NC command-specific reason codes**

| Value | Description | Comment |
|---|---|---|
| 0x4D01 | Abort Transfer | NC cannot proceed with transfer |
| 0x4D02 | Invalid Handle Value | Data Handle is invalid or not supported |
| 0x4D03 | Sequence count error | Chunk Number requested is not consecutive with the previous number transmitted |

### 8.4.116 Get Inventory Information command (`0x4E`)

The Get Inventory Information command may be used by the Management Controller to query the Network Controller for defined inventory information about the NC.

This command is defined as a package command.

Table 232 illustrates the packet format of the Inventory Information command.

**Table 232 – Get Inventory Information command packet format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

### 8.4.117 Get Inventory Information response (`0xCE`)

The package shall, in the absence of an error, always accept the Get Inventory Information command and send the response packet shown in Table 233. The value fields are defined as non-terminated ASCII strings except for the Manufacturing Timestamp which is timestamp104 as defined in DSP0240.

Currently no command-specific reason code is identified for this response.

**Table 233 – Get Inventory Information response packet format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..35** | Number of TLVs | Type-Length Field #1 | | Value Field #1 |
| | … | | | |
| | Checksum | | | |
| | Pad | | | |

#### 8.4.117.1 Inventory Information Type-Length-Value fields

The Type definitions for the inventory elements are defined below.

**Table 234 – Inventory Information Type-Length field**

| Bit Position | Field Description | Value Description |
|---|---|---|
| 7..0 | Attribute Name/Type | `0x0` = Manufacturer<br>`0x1` = Product / Model<br>`0x2` = Version<br>`0x3` = Part Number<br>`0x4` = Serial Number<br>`0x5` = Manufacturing Timestamp104<br>`0x6-0x7F` = Reserved<br>`0x80-0xAF` = Reserved for Manufacturer Use<br>`0xB0-0xFF` = Reserved for OEM use |
| 15..8 | Length | Length in bytes of the field |

### 8.4.118 OEM command `(0x50)`

The OEM command may be used by the Management Controller to request that the channel provide vendor-specific information. The Vendor Enterprise Number is the unique MIB/SNMP Private Enterprise number assigned by IANA per organization. Vendors are free to define their own internal data structures in the vendor data fields.

Table 235 illustrates the packet format of the OEM command.

**Table 235 – OEM command packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Manufacturer ID (IANA) | | | |
| **20…** | Vendor-Data<br>NOTE: The optional checksum is unspecified for the OEM command. OEMs supporting checksum validation for NC-SI commands may include the checksum in the OEM specific payload for the command and response. | | | |

### 8.4.119 OEM response (0xD0)

The channel shall return the “Unknown Command Type” reason code for any unrecognized enterprise number, using the packet format shown in Table 236. If the command is valid, the response, if any, is allowed to be vendor specific. The `0x8000` range is recommended for vendor-specific code.

Table 236 illustrates the packet format of the OEM command response.

**Table 236 – OEM response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Manufacturer ID (IANA) | | | |
| **24…** | Return Data (Optional) | | | |

### 8.4.120 PLDM Request (0x51)

The PLDM Request Packet may be used by the Management Controller to send PLDM commands over NC-SI/RBT. This command may be targeted at the entire package or a specific channel. It is expected that the MC will use PLDM Request command `0x51` to query the supported PLDM commands, before using Query Pending NC PLDM Request command.

Table 237 illustrates the packet format of the PLDM Request Packet over NC-SI/RBT.

**Table 237 – PLDM Request packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | PLDM Message Common Fields | | | |
| **20..** | PLDM Message Payload (zero or more bytes) + Payload Pad ) | | | |
| **..** | Checksum | | | |
| **..** | Pad | | | |

Refer to the PLDM Base specification (DSP0240) for details on the PLDM messaging control and discovery commands.

### 8.4.121 PLDM Response (0xD1)

The PLDM Response Packet may be used by the Network Controller to send PLDM responses over NC-SI/RBT. The package shall, in the absence of a checksum error or identifier mismatch, always accept the PLDM Request Command and send a response.

Table 238 illustrates the packet format of the PLDM command response.

**Table 238 – PLDM Response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | PLDM Message Common Fields | | | PLDM Completion Code |
| **24..** | PLDM Message Payload (zero or more bytes) + Payload Pad | | | |
| **..** | Checksum | | | |
| **..** | Ethernet Packet Pad | | | |

Refer to the PLDM Base specification (DSP0240) for details on the PLDM Response Messages.

Note that the NC-SI PLDM Response (`0xD1`) response/reason codes are only used to report the support, success, or failure of the PLDM Request command (`0x51`) at the NC-SI over RBT messaging layer. The PLDM Completion Code is used for determining the success or failure of the encapsulated PLDM Commands at the PLDM messaging layer.

### 8.4.122 Get Package UUID command (0x52)

The Get Package UUID command may be used by the Management Controller to query Universally Unique Identifier (UUID), also referred to as a globally unique ID (GUID), of the Network Controller over NC-SI/RBT. This command is targeted at the package. This command can be used by the MC to correlate endpoints used on different NC-SI transports (e.g., RBT, MCTP).

Table 239 illustrates the packet format of the Get Package UUID Command over NC-SI/RBT.

**Table 239 – Get Package UUID command packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

### 8.4.123 Get Package UUID response (0xD2)

The package shall, in the absence of an error, always accept the Get Package UUID command and send the response packet shown in Table 240. Currently no command-specific reason code is identified for this response.

**Table 240 – Get Package UUID response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Control Packet Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..35** | UUID bytes 1:16, respectively | | | |
| **36..39** | Checksum | | | |
| **40..45** | Pad | | | |

The individual fields within the UUID are stored most-significant byte (MSB) first per the convention described in RFC4122. RFC4122 specifies four different versions of UUID formats and generation algorithms suitable for use for a UUID. These are version 1 (`0001b`) "time based", and three "name-based" versions: version 3 (`0011b`) "MD5 hash", version 4 (`0100b`) "Pseudo-random", and version 5 "SHA1 hash". The version 1 format is recommended, however versions 3, 4, or 5 formats are also allowed to be used. See Table 241 for the UUID format version 1.

**Table 241 – UUID Format**

| Field | UUID Byte | MSB |
|---|---|---|
| time low | 1 | MSB |
| | 2 | |
| | 3 | |
| | 4 | |
| time mid | 5 | MSB |
| | 6 | |
| time high and version | 7 | MSB |
| | 8 | |
| clock seq and reserved | 9 | MSB |
| | 10 | |
| node | 11 | MSB |
| | 12 | |
| | 13 | |
| | 14 | |
| | 15 | |
| | 16 | |

### 8.4.124 Query and Set OEM AEN command (`0x54`)

The command Query and Set OEM AEN is used by the Management controller when sets of different OEM AENs, identified by the OEM's IANA value, are simultaneously supported by a NC. It allows the MC to query the channel or package for the active OEM AEN set as well as the other OEM AEN sets that are supported. The MC can then configure a particular IANA as the active one for subsequent issues of the AEN Enable command.

Implementation of this command is optional for those NCs that support only one set of OEM AENs

Implementation of this command is required when the NC has implemented multiple sets of OEM AENs and allows the MC to select a set that is different than the default

The NC may allow AENs from multiple sets to be simultaneously enabled through the successive uses of this command and AEN Enable

The NC shall interpret a null IANA in the received command as a request for the list of OEM AEN sets and shall not change the active set.

The Query and Set OEM AEN command is defined as a channel or a package command.

Table 242 illustrates the packet format of Query and Set OEM AEN command.

**Table 242 – Query and Set OEM AEN command packet**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | IANA Field | | | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

## 8.4.125 Query and Set OEM AEN Response (`0xD4`)

The Channel shall, in the absence of a checksum error or identifier mismatch, always accept the Query and Set OEM AEN Command and send a response.

For each supported OEM IANA, #1 through #n, three fields are required: the identifying IANA field, and the 16-bit Enabled AENs and Supported AENs fields that correspond 1:1 to bits 31..16 in the AEN Control Field of the AEN Enable command.

Table 243 illustrates the packet format of the Query and Set OEM AEN Response.

**Table 243 – Query and Set OEM AEN Response packet**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Reserved | Reserved | Reserved | # of IANAs |
| **24..27** | Configured IANA | | | |
| **28..31** | IANA # 1 | | | |
| **32..35** | IANA # 1 Enabled AENs | | IANA # 1 Supported AENs | |
| | IANA # 2 | | | |
| | … | | | |
| | Checksum | | | |
| | Pad | | | |

#### 8.4.125.1 # of IANAs field

An integer value representing the number of OEM AEN sets supported by the NC.

#### 8.4.125.2 Configured IANA field

The IANA representing the currently enabled OEM AEN set for configuration by subsequent Enable OEM AEN commands. If a valid IANA was sent in the command, the response shall confirm the change to that IANA set. If the sent IANA was not valid, the previously configured IANA set shall remain active.

#### 8.4.125.3 IANA #n field

The identifier for the n$^{th}$ OEM AEN set supported by the NC.

#### 8.4.125.4 IANA #n Enabled AENs field

A bitmap showing the currently enabled AENs from the IANA #n's set of supported AENs.

#### 8.4.125.5 IANA #n Supported AENs field

A bitmap showing the supported OEM AENs in the IANA #n's AEN set.

### 8.4.126 Transport-specific AEN Enable command (0x55)

Network Controller implementations shall support this command on the condition that the Network Controller generates one or more RBT-specific AENs defined in this specification or other NC-SI bindings such as DSP0261. The AEN Enable command enables and disables the different transport specific AENs supported by the Network Controller. The Network Controller shall copy the AEN MC ID field from the AEN Enable command into the MC ID field in every subsequent AEN sent to the Management Controller as defined in AEN Enable command.

Table 244 illustrates the packet format of the Enable Transport-specific AENs command.

**Table 244 – Transport-specific AEN Enable command packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Reserved | | Transport-specific AENs enable | |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

**Table 245 – Transport-specific AEN enable field format**

| Bit Position | Field Name | Value Description |
|---|---|---|
| 0 | Medium Change AEN Control (0x70) | 0b = Disable Medium Change AEN<br>1b = Enable Medium Change AEN<br>Relevant only for NC-SI/MCTP |
| 1 | Pending PLDM Request AEN (0x71) | 0b = Disable Pending PLDM Request AEN<br>1b = Enable Pending PLDM Request AEN |

| | | |
|---|---|---|
| | | Relevant only for PLDM over NC-SI control over RBT |
| 2 | Pending SPDM Request AEN (0x72) | 0b = Disable Pending SPDM Request AEN<br>1b = Enable Pending SPDM Request AEN<br>Relevant only for SPDM over NC-SI control over RBT |
| 3..15 | Reserved | Reserved |

### 8.4.127 Transport-specific AENs Enable Response (0xD5)

In the absence of any error, the package shall process and respond to the Transport-specific AEN Enable command by sending the response packet and payload shown in Table 246.

**Table 246 – Transport-specific AEN Enable Response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Checksum | | | |
| **…** | Pad | | | |

### 8.4.128 Query Pending NC PLDM Request (`0x56`)

The Query Pending NC PLDM Request may be used by the Management Controller to read the status of pending PLDM commands which the NC needs to send to the MC. Only one PLDM request can be handled by a Pending PLDM Request instance. When multiple requests are pending in the NC, each will be handled independently and the order at which requests are provided to the MC is decided by the NC.

Implementations using PLDM over RBT, where the NC has to send PLDM commands to the MC, shall support this command.

Table 247 illustrates the packet format of the Query Pending NC PLDM Request command.

**Table 247 – Query Pending NC PLDM Request packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

### 8.4.129 Query Pending NC PLDM Request Response (`0xD6`)

In the event there are no pending requests, the command shall execute successfully and return with no PLDM payload. Currently no command-specific reason code is identified for this response (see Table 248).

Table 248 illustrates the packet format of the Query Pending NC PLDM Request Response.

**Table 248 – Query Pending NC PLDM Request Response Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..** | PLDM Message Common Fields | | | PLDM Message Payload |
| | PLDM Message Payload + Payload Pad (zero or more bytes) | | | |
| | Checksum | | | |
| | Pad | | | |

**Table 249 – Query Pending NC PLDM Request Response parameters**

| Name | Meaning |
|---|---|
| PLDM Message Common fields | Optional, included only when there is a pending request |
| PLDM Message Payload | Optional, included only when there is a pending request |

### 8.4.130 Send NC PLDM Reply (`0x57`)

The Reply Pending PLDM command may be used by the Management Controller to provide the PLDM command response to previously read PLDM command from the NC that requires a response (Rq = 1, D = 0 in PLDM Message Common Fields). The response to this command further provides indication to the MC regarding additional pending PLDM NC commands.

Table 250 illustrates the packet format of the Send NC PLDM Reply command.

**Table 250 – Send NC PLDM Reply packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | PLDM Message Common Fields | | | PLDM Completion Code |
| **20..** | PLDM Message Payload (zero or more bytes) + Payload Pad | | | |
| | Checksum | | | |
| | Pad | | | |

### 8.4.131 Send NC PLDM Reply Response (`0xD7`)

Currently no command-specific reason code is identified for this response.

Table 251 illustrates the packet format of the Send NC PLDM Reply command.

**Table 251 – Send NC PLDM Reply Response packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Reserved | | | Flags |
| **24..27** | Checksum | | | |
| **28..45** | Pad | | | |

**Table 252 – Reply NC PLDM Response parameters**

| Name | Meaning |
|---|---|
| Flags bit 0 – Pending request | `0b` – No additional pending PLDM command from NC to MC |

| | |
|---|---|
| | `1b` – The NC has additional pending PLDM command to the MC |
| Flags bits 7:1 – Reserved | Reserved, always return 0. |

## 8.4.132 Get MC MAC Address command (`0x58`)

A network controller may provision MAC addresses for Out-Of-Band (OOB) management traffic. These MAC addresses are not visible to the host(s). Get MC MAC Address is used to discover MAC addresses provisioned on the network controller for the MC. Get MC MAC Address is a channel-specific command. For multiport devices, it is expected that the MC queries provisioned MC MAC Addresses on each channel individually.

Table 253 illustrates the packet format of the Get MC Address Command.

**Table 253 – Get MC MAC Address command packet format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

## 8.4.133 Get MC MAC Address response (`0xD8`)

In the response of Get MC MAC Address command, the network controller provides the information about the provisioned MAC address(es) for the MC on that channel. The NC shall, in the absence of an error, always accept the Get MC MAC Address command and send the response packet shown in Table 254. Currently no command-specific reason code is identified for this response.

**Table 254 – Get MC MAC Address response packet format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | Address Count | Reserved | | |
| **Variable** | Addr 1 Byte 5 | Addr 1 Byte 4 | Addr 1 Byte 3 | Addr 1 Byte 2 |
| | Addr 1 Byte 1 | Addr 1 Byte 0 | Addr 2 Byte 5 | Addr 2 Byte 4 |
| | … | | | |
| | … | | Pad (if needed) | |

### 8.4.133.1 Address Count

This field shall be set to the number of MC MAC addresses provisioned on the channel.

### 8.4.133.2 Reserved

This field shall be set to 0 by the network controller and shall be ignored by the management controller.

#### 8.4.133.3 Addr I Byte j

This field shall be set to the value of j^th byte (1 ≤ j ≤ 6) of i^th provisioned MC MAC address.

#### 8.4.133.4 Pad

If the number of MC MAC addresses is an odd number, then 2 bytes of the Pad field shall be present at the end of the payload to align the payload on a 32-bit boundary. If present, each byte of the Pad field shall be set to `0x00`.

If the number of MC MAC addresses is an even number, then 0 bytes of Pad shall be present.

### 8.4.134 SPDM command (`0x60`)

The SPDM command is used by the Management controller in RBT implementations to encapsulate and send a SPDM payload as defined in DSP0274 to the NC or alternately receive an encapsulated SPDM payload from the NC.

The SPDM payload must be smaller than the maximum NC-SI payload allowed over RBT. Payloads that exceed the RBT limits shall use SPDM's native multi-part transfer mechanism. Polling mode shall be used to transfer each part of a multi-part transfer from the NC.

The command response may be a long running command due to the nature of some SPDM tasks.

The SPDM command is defined as a package command.

This command and response are not supported on NC-SI over MCTP.

Table 255 illustrates the packet format of SPDM command.

**Table 255 – SPDM command packet**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | SPDM Version | Request Code | Param 1 | Param 2 |
| **20..** | SPDM Message Payload | | | |
| | Checksum | | | |
| | Pad | | | |

### 8.4.135 SPDM Response (`0xE0`)

The Package shall, in the absence of a checksum error or identifier mismatch, always accept the SPDM Command and send a response.

Table 256 illustrates the packet format of the SPDM Response.

**Table 256 – SPDM Response packet**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..23** | SPDM Version | Completion Code | Param 1 | Param 2 |
| **24..** | SPDM Response Payload | | | |
| | Checksum | | | |
| | Pad | | | |

## 8.4.136 Query Pending NC SPDM Request (0x61)

The Query Pending NC SPDM Request may be used by the Management Controller in RBT implementations to read the status of pending SPDM requests which the NC needs to send to the MC. Only one SPDM request can be handled by a Pending SPDM Request instance. When multiple requests are pending in the NC, each will be handled independently and the order at which requests are provided to the MC is decided by the NC.

The Query Pending NC SPDM command is defined as a package command.

This command and response are not supported on NC-SI over MCTP.

Table 257 illustrates the packet format of the Query Pending NC SPDM Request command.

**Table 257 – Query Pending NC SPDM Request packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Checksum | | | |
| **20..45** | Pad | | | |

## 8.4.137 Query Pending NC SPDM Request Response (0xE1)

In the event there are no pending requests, the command shall execute successfully and return with no SPDM payload. Currently no command-specific reason code is identified for this response (see Table 248).

Table 258 illustrates the packet format of the Query Pending NC SPDM Request Response.

**Table 258 – Query Pending NC SPDM Request Response Packet Format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Response Code | | Reason Code | |
| **20..** | SPDM Version | Request Code | Param 1 | Param 2 |

| | Bits | | | |
|---|---|---|---|---|
| Bytes | 31..24 | 23..16 | 15..08 | 07..00 |
| | SPDM Message Payload + Payload Pad (zero or more bytes) | | | |
| | Checksum | | | |
| | Pad | | | |

**Table 259 – Query Pending NC SPDM Request Response parameters**

| Name | Comment |
|---|---|
| SPDM Version | Optional, included only when there is a pending request |
| Request Code | Optional, included only when there is a pending request |
| Param1 | Optional, included only when there is a pending request |
| Param2 | Optional, included only when there is a pending request |
| SPDM Message Payload | Optional, included only when there is a pending request |

### 8.4.138 Send NC SPDM Reply (0x62)

The Reply Pending SPDM command may be used by the Management Controller to provide the SPDM command response to previously read SPDM command from the NC. The response to this command further provides indication to the MC regarding additional pending SPDM NC commands.

Table 260 illustrates the packet format of the Send NC SPDM Reply command.

**Table 260 – Send NC SPDM Reply packet format**

| | Bits | | | |
|---|---|---|---|---|
| Bytes | 31..24 | 23..16 | 15..08 | 07..00 |
| 00..15 | NC-SI Header | | | |
| 16..19 | SPDM Version | Completion Code | Param 1 | Param 2 |
| 20.. | SPDM Message Payload (zero or more bytes) + Payload Pad | | | |
| | Checksum | | | |
| | Pad | | | |

### 8.4.139 Send NC SPDM Reply Response (0xE2)

Currently no command-specific reason code is identified for this response.

Table 261 illustrates the packet format of the Send NC SPDM Reply command.

**Table 261 – Send NC SPDM Reply Response packet format**

| | Bits | | | |
|---|---|---|---|---|
| Bytes | 31..24 | 23..16 | 15..08 | 07..00 |
| 00..15 | NC-SI Header | | | |
| 16..19 | Response Code | | Reason Code | |

|  | Bits |  |  |  |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **20..23** | Reserved |  |  | Flags |
| **24..27** | Checksum |  |  |  |
| **28..45** | Pad |  |  |  |

**Table 262 – Reply NC SPDM Response parameters**

| Name | Meaning |
|---|---|
| Flags bit 0 – Pending request | `0b` – No additional pending SPDM command from NC to MC<br>`1b` – The NC has additional pending SPDM command to the MC |
| Flags bits 7:1–- Reserved | Reserved, always return 0. |

## 8.5 AEN packet formats

This clause defines the formats for the different types of AEN packets. For a list of the AEN types, see Table 18.

### 8.5.1 Link Status Change AEN

The Link Status Change AEN indicates to the Management Controller any changes in the channel's external Ethernet interface link status.

This AEN should be sent if any change occurred in the link status (that is, the actual link mode was changed). The Link Status and OEM Link Status fields reproduce the bit definitions defined in the Get Link Status Response Packet (see Table 51).

Table 263 illustrates the packet format of the Link Status Change AEN.

**Table 263 – Link Status Change AEN packet format**

|  | Bits |  |  |  |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | AEN Header |  |  |  |
| **16..19** | Reserved |  |  | AEN Type = `0x00` |
| **20..23** | Link Status |  |  |  |
| **24.27** | OEM Link Status |  |  |  |
| **28..31** | Checksum |  |  |  |

### 8.5.2 Configuration Required AEN

The Configuration Required AEN indicates to the Management Controller that the channel is transitioning into the Initial State. (This AEN is not sent if the channel enters the Initial State because of a Reset Channel command.)

NOTE: This AEN may not be generated in some situations in which the channel goes into the Initial State. For example, some types of hardware resets may not accommodate generating the AEN.

Table 264 illustrates the packet format of the Configuration Required AEN.

**Table 264 – Configuration Required AEN packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | AEN Header | | | |
| **16..19** | Reserved | | | AEN Type = 0x01 |
| **20..23** | Checksum | | | |

### 8.5.3 Host Network Controller Driver Status Change AEN

This AEN indicates a change of the Host Network Controller Driver Status. Table 265 illustrates the packet format of the AEN.

**Table 265 – Host Network Controller Driver Status Change AEN packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | AEN Header | | | |
| **16..19** | Reserved | | | AEN Type = `0x02` |
| **20..23** | Host Network Controller Driver Status | | | |
| **24..27** | Checksum | | | |

The Host Network Controller Driver Status field has the format shown in Table 266.

**Table 266 – Host Network Controller Driver Status format**

| Bit Position | Name | Description |
|---|---|---|
| 0 | Host Network Controller Driver Status | `0b` = The Network Controller driver for the host external network interface associated with this channel is not operational (not running).<br>`1b` = The Network Controller driver for the host external network interface associated with this channel is being reported as operational (running). |
| 1..31 | Reserved | Reserved |

## 8.5.4 Delayed Response Ready AEN

This AEN indicates the response to a delayed command is ready. Table 267 illustrates the packet format of the AEN.

NOTE: This AEN does not deliver the delayed command response, it must be retrieved separately.

**Table 267 – Delayed Response Ready AEN packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | AEN Header | | | |
| **16..19** | Reserved | | | AEN Type = `0x03` |
| **20..23** | Original Command Type | Original Command IID | Padding | |
| **24..27** | Checksum | | | |

The Original Command Type includes the Control Packet Type field of the completed command and the Original Command IID includes the IID field of the original command.

## 8.5.5 InfiniBand Link Status Change AEN

The InfiniBand Link Status Change AEN indicates to the Management Controller any changes in the channel's external InfiniBand interface link status.

This AEN should be sent if any change occurred in the IB link status (that is, the actual link mode was changed). The InfiniBand Link Status Change AEN specific fields reproduce the bit definitions defined in the Get IB Link Status Response Packet (see Table 206).

Table 273 illustrates the packet format of the InfiniBand Link Status Change AEN.

**Table 268 – InfiniBand Link Status Change AEN packet format**

<table>
<tr><th></th><th colspan="5">Bits</th></tr>
<tr><th>Bytes</th><th>31..24</th><th>23..16</th><th>15..08</th><th colspan="2">07..00</th></tr>
<tr><td>00..15</td><td colspan="5">AEN Header</td></tr>
<tr><td>16..19</td><td colspan="3">Reserved</td><td colspan="2">AEN Type = 0x04</td></tr>
<tr><td>20..23</td><td>IB Link Active Width</td><td>IB Link Supported Width</td><td>Link Type</td><td>Phys State</td><td>Logical Port State</td></tr>
<tr><td>24.27</td><td>Reserved</td><td>IB Link Active Speed</td><td>Reserved</td><td colspan="2">IB Link Supported Speed</td></tr>
<tr><td>28..31</td><td colspan="5">Checksum</td></tr>
</table>

## 8.5.6 Fibre Channel Link Status Change AEN

The Fibre Channel Link Status Change AEN indicates to the Management Controller any changes in the channel's external Fibre Channel interface link status including when trunked.

This AEN should be sent if any change occurred in the FC link status (that is, the actual link mode was changed). The Fibre Channel Link Status Change AEN specific fields reproduce the bit definitions defined in the Get FC Link Status Response Packet (see Table 182).

Table 279 illustrates the packet format of the FC Link Status Change AEN.

**Table 269 – Fibre Channel Link Status Change AEN packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | AEN Header | | | |
| **16..19** | Reserved | | | AEN Type = 0x05 |
| **20..23** | Num FC Ports | FC Trunk Status | FC Link Status | Trunk Speeds |
| **24.27** | Channel 1 Link Speed | Channel 2 Link Speed | Channel 3 Link Speed | Channel 4 Link Speed |
| **28..31** | Checksum | | | |

## 8.5.7 Transceiver Event AEN

This indicates to the Management Controller that a change in presence status or a thermal threshold in the SFF-compliant Transceiver attached to the channel has occurred.

Since some SFF cages have multiple TX and RX lanes, it is possible that multiple NC-SI channels are handled by a single transceiver module or copper cable assembly. In this case, subscribing to Transceiver Event AEN on one channel enables reporting of AENs for all such channels that are enabled for AENs. The NC shall send the Transceiver Event AEN on all affected channels that are enabled for AENs if one or more alerts are triggered.

In the case of FC port trunking (bonding), the 1:1 relationship of NC-SI channel to transceiver is lost and multiple transceivers will handle the aggregated traffic. When operating in the trunking mode, an enablement of this AEN on one channel will cover all transceivers that are members of the trunk. In this case, AENs can be generated individually for each member in the trunk by using the SFF Cage number field to identify the transceiver generating the AEN.

Table 270 illustrates the packet format of the AEN.

**Table 270 – Transceiver Event AEN packet format**

| | **Bits** | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | AEN Header | | | |
| **16..19** | Reserved | Transceiver Presence | SFF Cage Number | AEN Type = 0x06 |
| **20..23** | Transceiver Event List | | | |
| **24..27** | Reserved | | | |
| **28..31** | Checksum | | | |

### 8.5.7.1 SFF Cage Number field

SFF cage numbers are assigned to SFF cages in the implementation based on the NC-SI channel they are associated with (when not trunked) offset by one. Thus, the SFF cage associated with NC-SI channel 0 is #1, channel 1 has cage 2, etc.

### 8.5.7.2 Transceiver Event List field

The Transceiver Event List field has the format shown in Table 271.

**Table 271 – Transceiver Event List format**

| Bit Position | Name | Description |
|---|---|---|
| 0 | Low Temp Warning | 0b = no alert<br>1b = The Transceiver’s low temperature warning threshold has been exceeded |
| 1 | High Temp Warning | 0b = no alert<br>1b = The Transceiver’s high temperature warning threshold has been exceeded |
| 2 | Low Temp Alarm | 0b = no alert<br>1b = The Transceiver’s low temperature alarm threshold has been exceeded |
| 3 | High Temp Alarm | 0b = no alert<br>1b = The Transceiver’s high temperature alarm threshold has been exceeded |
| 4 | Low Voltage Warning | 0b = no alert<br>1b = The Transceiver’s low voltage warning threshold has been exceeded |
| 5 | High Voltage Warning | 0b = no alert<br>1b = The Transceiver’s high voltage warning |

| Bit Position | Name | Description |
|---|---|---|
| | | threshold has been exceeded |
| 6 | Low Voltage Alarm | 0b = no alert<br>1b = The Transceiver's low voltage alarm threshold has been exceeded |
| 7 | High Voltage Alarm | 0b = no alert<br>1b = The Transceiver's high voltage alarm threshold has been exceeded |
| 15..8 | 8 x RX Power Levels | 0b = no alert<br>1b = The Transceiver's RX Power alarm threshold has been exceeded. lsb is lane 1 thru msb is lane8 |
| 23..16 | 8 x TX Power Levels | 0b = no alert<br>1b = The Transceiver's TX Power alarm threshold has been exceeded. lsb is lane 1 thru msb is lane8 |
| 31..24 | 8 x TX Bias Levels | 0b = no alert<br>1b = The Transceiver's TX Bias Current alarm threshold has been exceeded. lsb is lane 1 thru msb is lane8 |
| | | |

#### 8.5.7.3 Transceiver Presence field

**Table 272 – Transceiver Presence format**

| Bit Position | Name | Description |
|---|---|---|
| 0 | Transceiver Presence Change | 0b = No change in presence detected<br>1b = The Transceiver was either removed or inserted. The insertion event reporting shall occur only after the Transceiver has completed its initialization stage |
| 7..1 | Reserved | |

### 8.5.8 Request Data Transfer AEN

This AEN indicates to the Management Controller that the NC is requesting the MC initiate a transfer of an opaque data package from the NC to the MC. It is sent using an Internal Channel ID value of 0x1F to indicate a package-level operation.

Table 273 illustrates the packet format of the AEN.

**Table 273 – Request Data Transfer AEN packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | AEN Header | | | |
| **16..19** | Reserved | | | AEN Type = `0x07` |
| **20..23** | Total Length of Transfer (Bytes) | | | |
| | Data Handle | | | |
| **24..27** | Checksum | | | |

### 8.5.9 Partition Link Status Change AEN

The Partition Link Status Change AEN indicates to the Management Controller any change in the internal link status of any partition on the channel. This AEN is only valid when the NC supports partitioning and it is enabled.

This AEN should be sent if any change occurred in the internal link status of any enabled partition on the channel.

Table 274 illustrates the packet format of the Partition Link Status Change AEN.

**Table 274 – Partition Link Status Change AEN packet format**

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | AEN Header | | | |
| **16..19** | Reserved | | | AEN Type = `0x08` |
| **20..23** | Reserved | | Partition Map | Link Status |
| **24.27** | Checksum | | | |

**Table 275 – Partition Map Field**

| Bit | Description |
|---|---|
| 0 | `0b` = Partition 1 on channel link state has not changed<br>`1b` = Partition 1 on channel link state has changed |
| 1 | `0b` = Partition 2 on channel link state has not changed<br>`1b` = Partition 2 on channel link state has changed |
| … | … |
| 7 | `0b` = Partition 8 on channel link state has not changed<br>`1b` = Partition 8 on channel link state has changed |

**Table 276 – Partition Link Status**

| Bit | Description |
|---|---|
| 0 | 0b = Partition 1 on channel link is down<br>1b = Partition 1 on channel link is up |
| 1 | 0b = Partition 2 on channel link is down<br>1b = Partition 2 on channel link is up |
| … | … |
| 7 | 0b = Partition 8 on channel link is down<br>1b = Partition 8 on channel link is up |

### 8.5.10 Thermal Shutdown Event AEN

The Thermal Shutdown Event AEN indicates to the Management Controller that NC device shutdown is imminent due to the defined thermal threshold being reached. It is sent using an Internal Channel ID value of 0x1F to indicate a package-level operation.

Table 277 illustrates the packet format of the Thermal Shutdown Event AEN.

**Table 277 – Thermal Shutdown Event AEN packet format**

| | Bits | | | |
|---|---|---|---|---|
| Bytes | 31..24 | 23..16 | 15..08 | 07..00 |
| 00..15 | AEN Header | | | |
| 16..19 | Reserved | | | AEN Type = 0x09 |
| 20..23 | Checksum | | | |

### 8.5.11 Pending PLDM Request AEN

The Pending PLDM Request AEN is an RBT-specific AEN used to alert the MC that there is a pending PLDM request for the MC in the NC. This AEN allows for the MC to poll for pending PLDM request on the NC at a lower rate. It is sent using an Internal Channel ID value of 0x1F to indicate a package-level operation.

As a transport-specific AEN, this AEN is enabled using the transport-specific AEN enable command and is controlled by bit 1 in Transport Specific AEN's enable field.

This AEN should be sent if there is a new pending PLDM request that is available in the NC designated to the MC, which was not reported to the MC through **Send NC PLDM Reply Response (**0xD7**)**. A Pending PLDM Request AEN should not be sent from the time the NC recognizes an incoming **Query Pending NC PLDM Request (0x56)** until the NC sends **Send NC PLDM Reply Response (**0xD7**)** for the PLDM request.

Table 278 – Pending PLDM Request AEN format

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Reserved | | | AEN Type = `0x71` |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

### 8.5.12 Pending SPDM Request AEN

The Pending SPDM Request AEN is an RBT-specific AEN used to alert the MC that there is a pending SPDM command request for the MC in the NC. It is sent using an Internal Channel ID value of `0x1F` to indicate a package-level operation.

As a transport-specific AEN, this AEN is enabled using the transport-specific AEN enable command and is controlled by bit 2 in Transport Specific AEN's enable field.

This AEN should be sent if there is a new pending SPDM request that is generated in the NC designated for the MC, which was not reported to the MC through Send NC SPDM Reply Response (`0xE2`). A Pending SPDM Request AEN should not be sent from the time the NC recognizes an incoming Query Pending NC SPDM Request `(0x61)` until the NC sends Send NC SPDM Reply Response (`0xE2`) for the SPDM request.

Table 278 – Pending SPDM Request AEN format

| | Bits | | | |
|---|---|---|---|---|
| **Bytes** | **31..24** | **23..16** | **15..08** | **07..00** |
| **00..15** | NC-SI Header | | | |
| **16..19** | Reserved | | | AEN Type = `0x72` |
| **20..23** | Checksum | | | |
| **24..45** | Pad | | | |

# 9 Packet-based and opcode timing

Table 279 presents the timing specifications for a variety of packet-to-electrical-buffer interactions, inter-packet timings, and opcode processing requirements. The following timing parameters shall apply to NC-SI over RBT binding defined in this specification.

**Table 279 – NC-SI packet-based and opcode timing parameters**

| Name | Symbol | Value | Description |
|---|---|---|---|
| Package Deselect to Hi-Z Interval | T1 | 200 µs, max | Maximum time interval from when a Network Controller completes transmitting the response to a Deselect Package command to when the Network Controller outputs are in the high-impedance state<br>Measured from the rising edge of the first clock that follows the last bit of the packet to when the output is in the high-impedance state as defined in clause 10 |
| Package Output to Data | T2 | 2 clocks, min | Minimum time interval after powering up the output drivers before a Network Controller starts transmitting a packet through the NC-SI interface<br>Measured from the rising edge of the first clock of the packet |
| Network Controller Power Up Ready Interval | T4 | 2 s, max | Time interval from when the NC-SI on a Network Controller is powered up to when the Network Controller is able to respond to commands over the NC-SI<br>Measured from when $V_{ref}$ becomes available |
| Normal Execution Interval | T5 | 50 ms, max | Maximum time interval from when a controller receives a command to when it delivers a response to that command, unless otherwise specified<br>Measured from the rising edge of the first clock following the last bit of the command packet to the rising edge of the clock for the first bit of the response packet |
| Asynchronous Reset Interval | T6 | 2 s, max | Interval during which a controller may not recognize or respond to commands or handle Pass-through traffic due to an Asynchronous Reset event. See clause 6.1.8<br>For a Management Controller, this means that a Network Controller could become unresponsive for up to T6 seconds if an Asynchronous Reset event occurs. This is not an error condition. The Management Controller retry behavior should be designed to accommodate this possibility. |
| Synchronous Reset Interval | T7 | 2 s, max | Interval during which a controller may not recognize or respond to commands or handle Pass-through traffic due to a Synchronous Reset event. See clause 6.1.8<br>Measured from the rising edge of the first clock following the last bit of the Reset Channel response packet |
| Token Timeout | T8 | 32,000 REF_CLK min | Number of REF_CLKs before timing out while waiting for a TOKEN to be received |

| Name | Symbol | Value | Description |
|---|---|---|---|
| Opcode Processing | T9 | 32 REF_CLK max | Number of REF_CLKs after receiving an opcode on ARB_IN to decode the opcode and generate the next opcode on ARB_OUT<br>Measured from the falling edge of the last bit of the opcode received on ARB_IN to the rising edge of the next opcode on ARB_OUT |
| Opcode Bypass Delay | T10 | 32 REF_CLK max | Number of REF_CLK delays between a bit received on ARB_IN and the corresponding bit passed on to ARB_OUT while in Bypass Mode<br>Measured from the falling edge of the last bit of the opcode received on ARB_IN to the rising edge of the next opcode on ARB_OUT |
| TOKEN to RXD | T11 | T2 min, 32 REF_CLK max | Number of REF_CLKs after receiving TOKEN to when packet data is driven onto the RXD lines<br>Measured from the falling edge of the last bit of the opcode received on ARB_IN to the rising edge of the first clock of the next packet on RXD |
| Max XOFF Renewal Interval | T12 | 50,331,648 REF_CLK max | Maximum time period (3 XOFF Frame timer cycles) during which a channel within a package is allowed to request and renew a single XOFF condition after requesting the initial XOFF |
| IPG to TOKEN Opcode Overlap | T13 | 6 REF_CLK max | Maximum number of REF_CLKs that the beginning of TOKEN transmission can precede the end of the Inter Packet Gap. For more information, see clause 7.3.8. |
| Delayed Execution Interval | T14 | 4 s, max | Maximum time interval from when a controller receives a command to when it delivers a response to that command, including all responses with “Delayed Response” code<br>Measured from the rising edge of the first clock following the last bit of the command packet to the rising edge of the clock for “Delayed Response Ready” AEN if enabled or to the moment the NC is internally ready with a response for a polling command. |

NOTE: If hardware arbitration is in effect, the hardware arbitration output buffer enable/disable timing specifications take precedence.

# 10 RBT Electrical specification

This clause provides background information about the NC-SI RBT specification, describes the RBT topology, and defines the electrical, timing, signal behavior, and power-up characteristics for the RBT physical interface.

## 10.1 Topologies

The electrical specification defines the RBT electrical characteristics for one management processor and one to four Network Controller packages in a bussed "multi-drop" arrangement. The actual number of devices that can be supported may differ based on the trace characteristics and routing used to interconnect devices in an implementation.

Figure 16 shows an example topology.

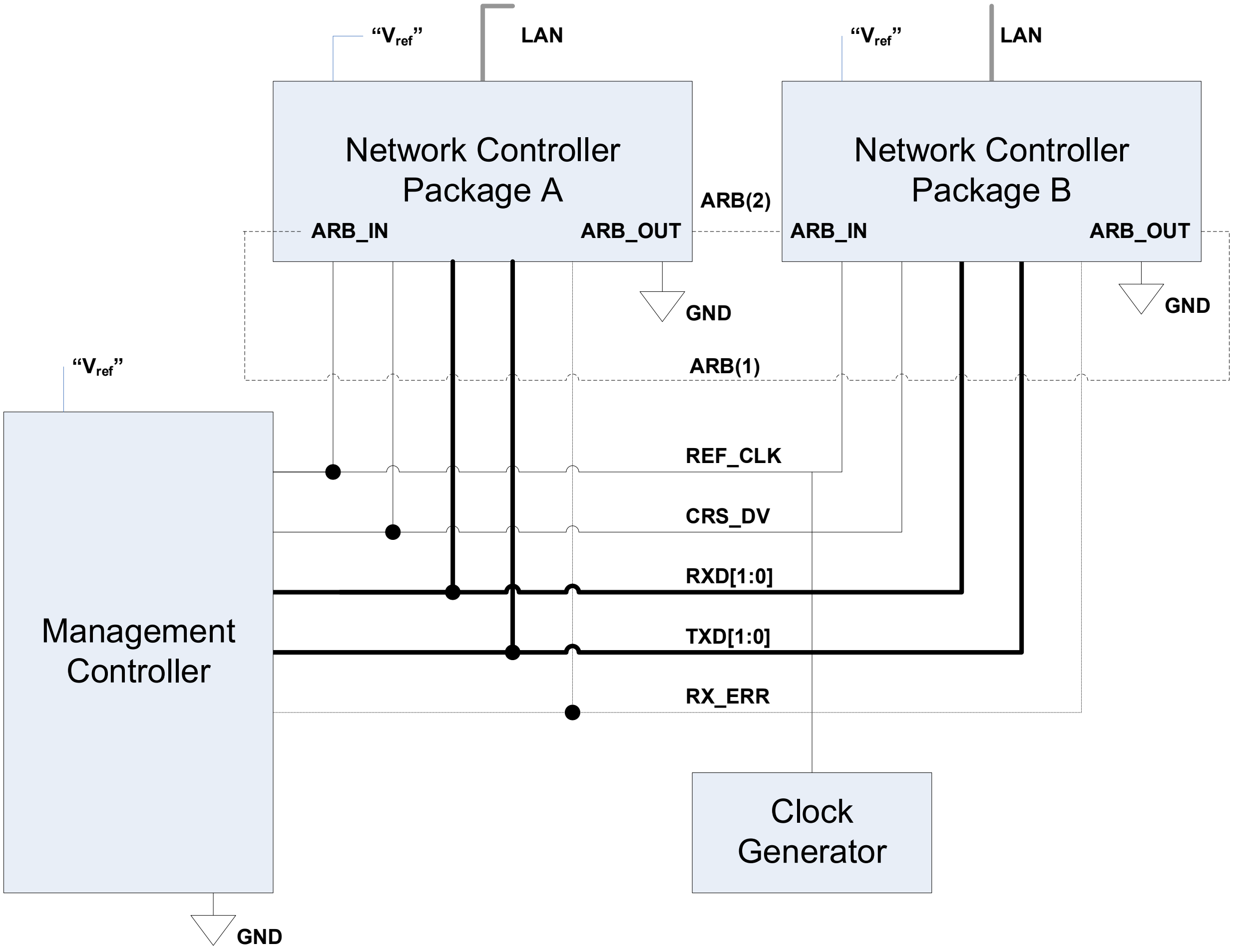


**Figure 16 – Example NC-SI RBT signal interconnect topology**

## 10.2 Electrical and signal characteristics and requirements

This clause defines the electrical, timing, signal behavior, and power-up characteristics for the NC-SI RBT physical interface.

### 10.2.1 Companion specifications

Implementations of the physical interface and signaling for RBT shall meet the specifications in RMII and IEEE 802.3, except where those requirements differ or are extended with specifications provided in this document, in which case the specifications in this document shall take precedence.

### 10.2.2 Full-duplex operation

RBT is specified only for full-duplex operation. Half-duplex operation is not covered by this specification.

### 10.2.3 Signals

Table 280 lists the signals that make up the RBT physical interface.

Unless otherwise specified, the high level of a RBT signal corresponds to its asserted state, and the low level represents the de-asserted state. For data bits, the high level represents a binary '1' and the low level a binary '0'.

**Table 280 – Physical RBT signals**

| Signal Name | Direction (with respect to the Network Controller) | Direction (with respect to the Management Controller MAC) | Use | Mandatory or Optional |
|---|---|---|---|---|
| REF_CLK [a] | Input | Input | Clock reference for receive, transmit, and control interface | M |
| CRS_DV [b] | Output | Input | Carrier Sense/Receive Data Valid | M |
| RXD[1:0] | Output | Input | Receive data | M |
| TX_EN | Input | Output | Transmit enable | M |
| TXD[1:0] | Input | Output | Transmit data | M |
| RX_ER | Output | Input | Receive error | O |
| ARB_IN | Input [c] | N/A | Network Controller hardware arbitration Input | O [c] |
| ARB_OUT | Output [c] | N/A | Network Controller hardware arbitration Output | O [c] |

A device can provide an additional option to allow it to be configured as the source of REF_CLK, in which case the device is not required to provide a separate REF_CLK input line, but it can use REF_CLK input pin as an output. The selected configuration shall be in effect at NC power up and remain in effect while the NC is powered up.

In the *RMII Specification*, the MII Carrier Sense signal, CRS, was combined with RX_DV to form the CRS_DV signal. When RBT is using its specified full-duplex operation, the CRS aspect of the signal is not required; therefore, the signal shall provide only the functionality of RX_DV as defined in IEEE 802.3. (This is equivalent to the CRS_DV signal states in *RMII Specification* when a carrier is constantly present.) The Carrier Sense aspect of the CRS_DV signal is not typically applicable to RBT because it does not typically detect an actual carrier (unlike an actual PHY). However, the Network Controller should emulate a carrier-present status on CRS_DV per IEEE 802.3 in order to support Management Controller MACs that may require a carrier-present status for operation.

If hardware arbitration is implemented, the Network Controller package shall provide both ARB_IN and ARB_OUT connections. In some implementations, ARB_IN may be required to be tied to a logic high or low level if it is not used.

### 10.2.4 High-impedance control

Shared RBT operation requires Network Controller devices to be able to set their outputs (RXD[1:0], CRS_DV, and, if implemented, RX_ER) into a high-impedance state either upon receipt of a command being received, or, if hardware-based arbitration is enabled as a result of hardware-based arbitration. A pull-down resistor should be provided on high impedance signals to prevent them from floating when not driven.

Network Controllers shall leave their RBT outputs in the high-impedance state on interface power up and shall not drive them until the package is selected. For additional information about Network Controller packages, see 8.4.5.

For RBT output signals in this specification, unless otherwise specified, the high-impedance state is defined as the state in which the signal leakage meets the $I_Z$ specification provided in 10.2.5.

### 10.2.5 Hardware Implementations

A variety of shared RBT hardware implementations are possible, in such cases the designer must take care to ensure the HW arbitration loop is maintained when used, even if some RBT devices are not present. Pull resistors are recommended to be place on the system board side of any connector for add-in RBT cards so that a proper resistance for the high impedance signals can be maintained.

### 10.2.6 DC characteristics

This clause defines the DC characteristics of the RBT physical interface.

#### 10.2.6.1 Signal levels

CMOS 3.3 V signal levels are used for this specification.

The following characteristics apply to DC signals:

- Unless otherwise specified, DC signal levels and $V_{ref}$ are measured relative to Ground (GND) at the respective device providing the interface, as shown in Figure 17.
- Input specifications refer to the signals that a device shall accept for its input signals, as measured at the device.
- Output specifications refer to signal specifications that a device shall emit for its output signals, as measured at the device.

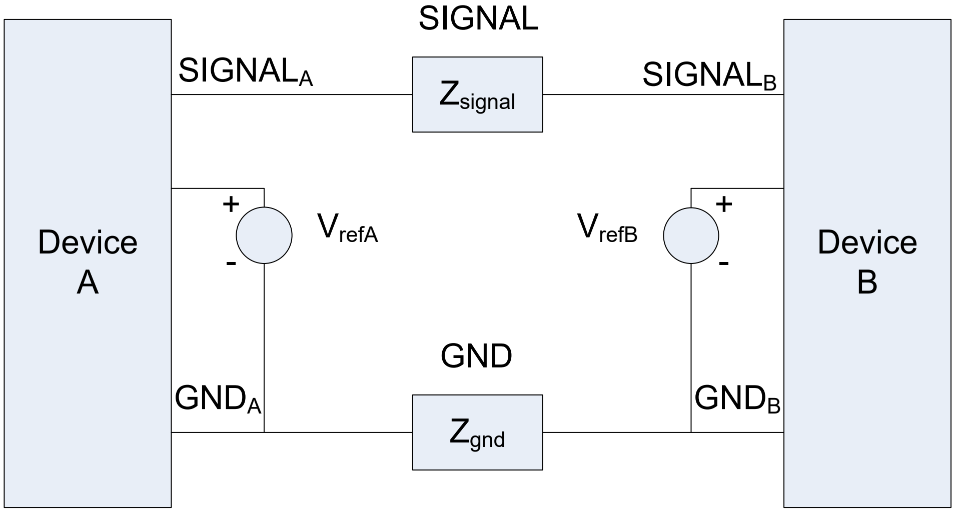


**Figure 17 – DC measurements**

Table 281 provides DC specifications.

**Table 281 – DC specifications**

| Parameter | Symbol | Conditions | Minimum | Typical | Maximum | Units |
|---|---|---|---|---|---|---|
| IO reference voltage | $V_{ref}$ [a] | | 3.0 | 3.3 | 3.6 | V |
| Signal voltage range | $V_{abs}$ | | -0.300 | | 3.765 | V |
| Input low voltage | $V_{il}$ | | | | 0.8 | V |
| Input high voltage | $V_{ih}$ | | 2.0 | | | V |
| Input high current | $I_{ih}$ | $V_{in} = V_{ref} = V_{ref,max}$ | 0 | | 200 | µA |
| Input low current | $I_{il}$ | $V_{in}$ = 0 V | -20 | | 0 | µA |
| Output low voltage | $V_{ol}$ | $I_{ol}$ = 4 mA, $V_{ref}$ = min | 0 | | 400 | mV |
| Output high voltage | $V_{oh}$ | $I_{oh}$ = -4 mA, $V_{ref}$ = min | 2.4 | | $V_{ref}$ | V |
| Clock midpoint reference level | $V_{ckm}$ | | | | 1.4 | V |
| Leakage current for output signals in high-impedance state | $I_z$ | $0 \le V_{in} \le V_{ref}$<br>at $V_{ref} = V_{ref,max}$ | -20 | | 20 | µA |

$V_{ref}$ = Bus high reference level (typically the NC-SI logic supply voltage). This parameter replaces the term supply voltage because actual devices may have internal mechanisms that determine the operating reference for RBT that are different from the devices' overall power supply inputs.

$V_{ref}$ is a reference point that is used for measuring parameters (such as overshoot and undershoot) and for determining limits on signal levels that are generated by a device. To facilitate system implementations, a device shall provide a mechanism (for example, a power supply pin, internal programmable reference, or reference level pin) to allow $V_{ref}$ to be set to within 20 mV of any point in the specified $V_{ref}$ range. This approach enables a system integrator to establish an interoperable $V_{ref}$ level for devices on RBT.

## 10.2.7 AC characteristics

This clause defines the AC characteristics of the RBT physical interface.

### 10.2.7.1 Rise and fall time measurement

Rise and fall time are measured between points that cross 10% and 90% of $V_{ref}$ (see Table 281). The middle points (50% of $V_{ref}$) are marked as $V_{ckm}$ and $V_m$ for clock and data, respectively.

### 10.2.7.2 REF_CLK measuring points

In Figure 18, REF_CLK duty cycle measurements are made from $V_{ckm}$ to $V_{ckm}$. Clock skew $T_{skew}$ is measured from $V_{ckm}$ to $V_{ckm}$ of two RBT devices and represents the maximum clock skew between any two devices in the system.

### 10.2.7.3 Data, control, and status signal measuring points

In Figure 18, all timing measurements are made between $V_{ckm}$ and $V_m$. $T_{co}$ is measured with a capacitive load between 10 pF and 50 pF. Propagation delay $T_{prop}$ is measured from $V_m$ on the transmitter to $V_m$ on the receiver.

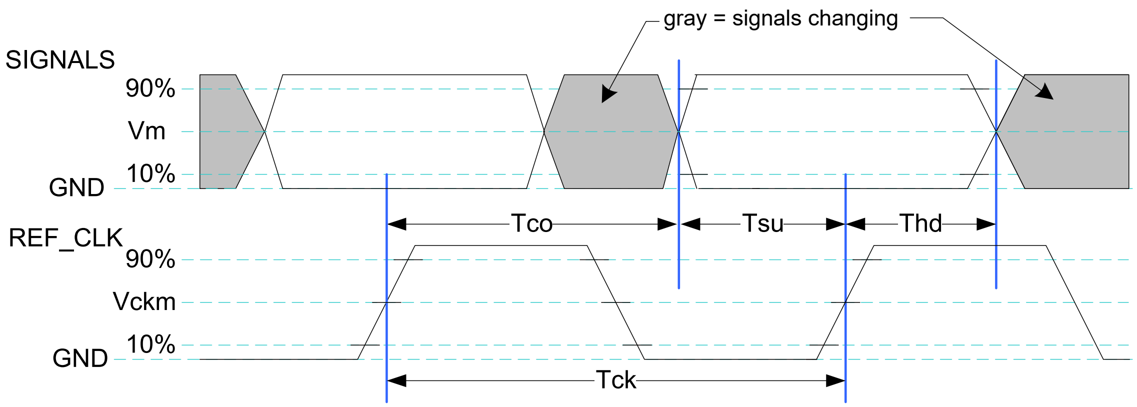


**Figure 18 – AC measurements**

Table 282 provides AC specifications.

**Table 282 – AC specifications**

| Parameter | Symbol | Minimum | Typical | Maximum | Units |
|---|---|---|---|---|---|
| REF_CLK Frequency | | | 50 | 50+100 ppm | MHz |
| REF_CLK Duty Cycle | | 35 | | 65 | % |
| Clock-to-out [a]<br>(10 pF ≤ $C_{load}$ ≤ 50 pF) | $T_{co}$ | 2.5 | | 12.5 | ns |
| Skew between clocks | $T_{skew}$ | | | 1.5 | ns |
| TXD[1:0], TX_EN, RXD[1:0], CRS_DV, RX_ER, and ARB_IN data setup to REF_CLK rising edge | $T_{su}$ | 3 | | | ns |
| TXD[1:0], TX_EN, RXD[1:0], CRS_DV, RX_ER, and ARB_OUT data hold from REF_CLK rising edge | $T_{hd}$ | 1 | | | ns |
| Signal Rise/Fall Time | $T_r/T_f$ | 0.5 | | 6 | ns |
| REF_CLK Rise/Fall Time | $T_{ckr}/T_{ckf}$ | 0.5 | | 3.5 | ns |
| Interface Power-Up High-Impedance Interval | $T_{pwrz}$ | 2 | | | µs |
| Power Up Transient Interval (recommendation) | $T_{pwrt}$ | | | 100 | ns |
| Power Up Transient Level (recommendation) | $V_{pwrt}$ | -200 | | 200 | mV |
| | | | | | |
| REF_CLK Startup Interval | $T_{clkstrt}$ | | | 100 | ms |

This timing relates to the output pins, while $T_{su}$ and $T_{hd}$ relate to timing at the input pins.

### 10.2.7.4 Timing calculation (informative)

#### 10.2.7.4.1 Setup time calculation

$$T_{su} \leq T_{clk} - (T_{skew} + T_{co} + T_{prop})$$

#### 10.2.7.4.2 Hold time calculation

$$T_{hd} \le T_{co} - T_{skew} + T_{prop}$$

### 10.2.7.5 Overshoot specification

Devices shall accept signal overshoot within the ranges specified in Figure 19, measured at the device, without malfunctioning.

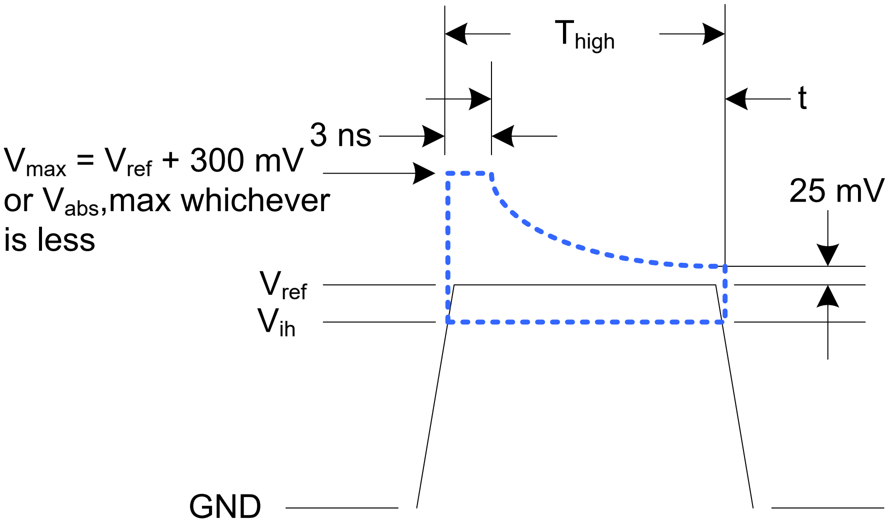

$T_{high}$
t
3 ns
$V_{max} = V_{ref}$ + 300 mV or $V_{abs}$,max whichever is less
25 mV
$V_{ref}$
$V_{ih}$
GND

**Figure 19 – Overshoot measurement**

The signal may overshoot up to the specified $V_{max}$ for the first 3 ns following the transition above $V_{ih}$. Following that interval is an exponential decay envelope equal to the following:

$$V_{ref} + V_{os} * e^\wedge[- K * ( [t - 3 \text{ ns}] /T_{d})]$$

Where, for t = 3 to 10 ns:

t = 0 corresponds to the leading crossing of $V_{ih}$, going high.

$V_{ref}$ is the bus high reference voltage (see 10.2.5).

$V_{abs}$,max is the maximum allowed signal voltage level (see 10.2.5).

$V_{os} = V_{max} - V_{ref}$

$K = l_{n}(25 \text{ mV}/V_{os})$

$T_{d} = 7$ ns

For t > 10 ns, the $V_{ref}$ + 25 mV limit holds flat until the conclusion of $T_{high}$.

### 10.2.7.6 Undershoot specification

Devices are required to accept signal undershoot within the ranges specified in Figure 20, measured at the device, without malfunctioning.

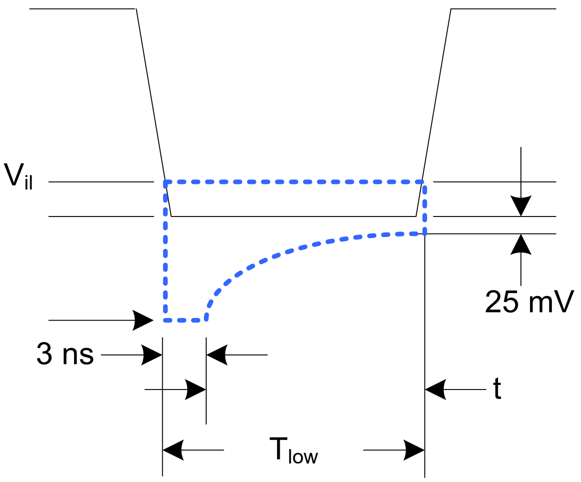


**Figure 20 – Undershoot measurement**

The signal is allowed to undershoot up to the specified $V_{abs}$,min for the first 3 ns following the transition above $V_{il}$. Following that interval is an exponential envelope equal to the following:

* ([t –3 ns]/$T_d$)]

Where, for t = 3 to 10 ns:

t = 0 corresponds to the leading crossing of $V_{il}$, going low.

$V_{abs}$,min is the minimum allowed signal voltage level (see 10.2.5).

K = $l_n$(25 mV/$V_{os}$)

$T_d$ = 7 ns

For t > 7 ns, the GND – 25 mV limit holds flat until the conclusion of $T_{low}$.

## 10.2.8 Interface power-up

To prevent signals from back-powering unpowered devices, it is necessary to specify a time interval during which signals are not to be driven until devices sharing the interface have had time to power up. To facilitate system implementation, the start of this interval shall be synchronized by an external signal across devices.

### 10.2.8.1 Power-up control mechanisms

The device that provides the interface shall provide one or more of the following mechanisms to enable the system integrator to synchronize interface power-up among devices on the interface:

- **Device power supply pin**

  The device has a power supply pin that the system integrator can use to control power-up of the interface. The device shall hold its outputs in a high-impedance state (current < $I_z$) for at least $T_{pwrz}$ seconds after the power supply has initially reached its operating level (where the power supply operating level is specified by the device manufacturer).

- **Device reset pin or another similar signal**

  The device has a reset pin or other signal that the system integrator can use to control the power-up of the interface. This signal shall be able to be driven asserted during interface power-up and de-asserted afterward. The device shall hold its outputs in a high-impedance state (current < $I_z$) for at least $T_{pwrz}$ seconds after the signal has been de-asserted, other than as described in clause 10.2.8.2. It is highly recommended that a single signal be used; however, an implementation is allowed to use a combination of signals if required. Logic levels for the signals are as specified by the device manufacturer.

- **REF_CLK detection**

  The device can elect to detect the presence of an active REF_CLK and use that for determining whether NC-SI power up has occurred. It is recommended that the device should count at least 100 clocks and continue to hold its outputs in a high-impedance state (current < $I_z$) for at least $T_{pwrz}$ seconds more (Informational: 100 clocks at 50 MHz is 2 us).

#### 10.2.8.2 Power-up transients

It is possible that a device may briefly drive its outputs while the interface or device is first receiving power, due to ramping of the power supply and design of its I/O buffers. It is recommended that devices be designed so that such transients, if present, are less than $V_{pwrt}$ and last for no more than $T_{pwrt}$.

### 10.2.9 REF_CLK startup

REF_CLK shall start up, run, and meet all associated AC and DC specifications within $T_{clkstrt}$ seconds of interface power up.

## 10.3 RBT Implementation guidance

This specification does not define implementation requirements due to the wide variation in architectures, devices and materials used. Following good engineering practices are a key part of a successful NC-SI RBT implementation:

- Care must be taken in placement and layout
- Do a complete signal integrity analysis including determining what, if any, termination is required
- Minimize stubs
- Have uniform clock trace lengths
- Minimize noise on high-impedance signals

# ANNEX A
# (normative)

# Extending the model

This annex explains how the model can be extended to include vendor-specific content.

## Commands extension

A Network Controller vendor can implement extensions and expose them using OEM commands, as described in clause 8.4.118.

## Design considerations

This clause describes certain design considerations for vendors of Management Controllers.

### PHY support

Although not a requirement of this specification, a Management Controller vendor can design the RBT interface in such a manner that it could also be configured for use with a conventional RMII PHY. This would enable the vendor’s controller to also be used in applications where a direct, non-shared network connection is available or preferred for manageability.

### Multiple Management Controllers support

Currently, there is no requirement for Management Controllers to be able to put their TXD output lines and other output lines into a high-impedance state, because the present definition assumes only one Management Controller on the bus. However, component vendors can provide such control capabilities in their devices to support possible future system topologies where more than one Management Controller shares the bus to enable functions such as Management Controller fail-over or to enable topologies where more than one Management Controller can participate in NC-SI communications on the bus. If a vendor elects to make such provision, it is recommended that the TXD line and the remaining output lines be independently and dynamically switched between a high-impedance state and re-enabled under firmware control.

# ANNEX B
# (informative)

# Relationship to RMII Specification

## Differences with the *RMII Specification*

The following list presents key differences and clarifications between the *NC-SI Specification* and sections in the *RMII Specification*. (Section numbers refer to the *RMII Specification*.)

- General: Where specifications from IEEE 802.3 apply, this specification uses the version specified in clause 2 (Normative references), rather than the earlier IEEE 802.3u version that is referenced by RMII.
- Section 1.0:
  - The *NC-SI Specification* requires 100 Mbps support, but it does not specify a required minimum. (10 Mbps support is not required by NC-SI.)
  - Item 4. (Signals may or may not be considered to be TTL. NC-SI is not 5-V tolerant.)
- Section 2.0:
  - Comment: NC-SI chip-to-chip includes considerations for multi-drop and allows for non-PCB implementations and connectors (that is, not strictly point-to-point).
- Section 3.0:
  - Note/Advisory: The NC-SI clock is provided externally. An implementation can have REF_CLK provided by one of the devices on the bus or by a separate device.
- Section 5.0:
  - For NC-SI, the term *PHY* is replaced by *Network Controller*.
- Table 1:
  - The information in Table 1 in the *RMII Specification* is superseded by tables in this specification.
- Section 5.1, paragraph 2:
  - The *NC-SI Specification* allows 100 ppm. This supersedes the *RMII Specification*, which allows 50 ppm.
- Section 5.1, paragraph 3:
  - The NC-SI inherits the same requirements. The NC-SI MTU is required only to support Ethernet MTU with VLAN, as defined in the IEEE 802.3 version listed in clause 2
  - Section 5.1 paragraph 4:
  - The *RMII Specification* states: "During a false carrier event, CRS_DV shall remain asserted for the duration of carrier activity." This statement is not applicable to full-duplex operation of the NC-SI. CRS_DV from the Network Controller is used only as a data valid (DV) signal. Because the Carrier Sense aspect of CRS_DV is not used for full-duplex operation of the NC-SI, the Network Controller would not generate false carrier events for the NC-SI. However, it is recommended that the MAC in the Management Controller be able to correctly detect and handle these patterns if they occur, as this would be part of enabling the Management Controller MAC to also be able to work with an RMII PHY.

- Section 5.2:
  - The NC-SI does not specify a 10 Mbps mode. The Carrier Sense aspect of CRS_DV is not used for full-duplex operation of NC-SI.
- Section 5.3.1:
  - While the NC-SI does not specify Carrier Sense usage of CRS_DV, it is recommended that a Management Controller allow for CRS_DV toggling, in which CRS_DV toggles at 1/2 clock frequency, and that Management Controller MACs tolerate this and realign bit boundaries correctly in order to be able to work with an RMII PHY also.
- Section 5.3.2:
  - There is no 10 Mbps mode specified for the NC-SI RBT interface.
- Section 5.3.3:
  - Generally, there is no expectation that the Network Controller will generate these error conditions for the NC-SI; however, the MAC in the Management Controller should be able to correctly detect and handle these patterns if they occur.
- Section 5.3.3:
  - The NC-SI does not specify or require support for RMII Registers.
- Section 5.5.2:
  - Ignore (N/A) text regarding 10 Mbps mode. RBT does not specify or require interface operation in 10 Mbps mode.
- Section 5.6:
  - The Network Controller will not generate collision patterns for the specified full-duplex operation of the NC-SI; however, the MAC in the Management Controller should be able to detect and handle these patterns if they occur in order to be able to work with an RMII PHY also.
- Section 5.7:
  - NC-SI RBT uses the IEEE 802.3 version listed in clause 2 instead of 802.3u as a reference.
- Section 5.8:
  - Loopback operation is not specified for the NC-SI RBT interface.
- Section 7.0:
  - The NC-SI RBT electrical specifications (clause 10) take precedence. (For example, section 7.4.1 in the *RMII Specification* for capacitance is superseded by *NC-SI Specification* 25 pF and 50 pF target specifications.)
- Section 8.0:
  - NC-SI RBT uses the IEEE 802.3 version listed in clause 2 (Normative references) as a reference, instead of 802.3u.

# ANNEX C
# (informative)

# Change log

| Version | Date | Description |
|---|---|---|
| 1.0.0 | 2009-07-21 | DMTF Standard release |
| 1.0.1 | 2013-01-24 | DMTF Standard release |
| 1.1.0 | 2015-09-23 | DMTF Standard release |
| 1.1.1 | 2021-04-13 | Updated to comply with ISO guidelines |
| 1.2.0b | 2019-08-19 | DMTF Work in Progress release |
| 1.2.0WIP80 | 2021-08-25 | DMTF Work in Progress release |
| 1.2WIP90 | 2022-06-03 | DMTF Work in Progress release |
| 1.2.0WIP95 | 2022-09-01 | DMTF Work in Progress release |
| 1.2.0WIP99 | 2023-06-02 | DMTF Work in Progress release |
| 1.2.0 | TBD | DMTF Standard release |

# Bibliography

IANA, Internet Assigned Numbers Authority (https://www.iana.org/). A body that manages and organizes numbers associated with various Internet protocols.

DMTF DSP4014, *DMTF Process for Working Bodies* 2.2, August 2015
https://www.dmtf.org/sites/default/files/standards/documents/DSP4014_2.2.0.pdf