Verification Script Engine

for

Teledyne LeCroy

PCIe Protocol Analysis™

Reference Manual

For PCIe Protocol Analysis version 13.32

Generated: 4/28/2026 12:22 PM

Document Disclaimer

The information contained in this document has been carefully checked and is believed to be reliable. However, no responsibility can be assumed for inaccuracies that may not have been detected.

Teledyne LeCroy reserves the right to revise the information presented in this document without notice or penalty.

Trademarks and Servicemarks

Teledyne LeCroy, *CATC Trace, PETracer, PCIe Protocol Suite, PCIe Protocol Analysis, Summit, Summit T3-16, Summit T3-8, Summit T34, Summit T28, Summit T24, Summit Z3-16, Summit Z416, and BusEngine* are trademarks of Teledyne LeCroy.

Microsoft and *Windows* are registered trademarks of Microsoft Inc.

All other trademarks are property of their respective companies.

Copyright

Table of Contents

1 Introduction

This document describes the Teledyne LeCroy Verification Script Engine (VSE), a utility in the PCIe Protocol Suite™ software. The VSE allows users to perform custom analyses of PCI Express™ (PE) traffic, recorded using the new generation of PCI Express protocol analyzers.

VSE allows users to ask the PCIe Protocol Suite application to send some desired “events” (currently defined as packets, link transactions, split transactions, AHCI, ATA, NVM transactions or commands) from a PE trace to a verification script written using File-Based Decoding. This script then evaluates the sequence of events (timing, data or both) in accordance with user-defined conditions and performs post-processing tasks; such as exporting key information to external text-based files or sending special Automation/COM notifications to user client applications.

VSE allows users to easily retrieve information about any field in a PE packet header or link/split/NVM/AHCI/ATA transaction or command, and to make complex timing calculations between different events in a pre-recorded trace. It also allows filtering-in or filtering-out of data with dynamically changing filtering conditions, porting of information to a special output window, saving of data to text files, and sending of data to COM clients connected to a PCIe Protocol Suite application.

Note: All scripting conventions apply to both the Z3-16 and the Z416.

2 Verification Script Structure

Writing a verification script is easy, as long as you follow a few rules and have some understanding of how the PCIe Protocol Suite™ application interacts with running scripts.

The main script file that contains the text of the verification script should have extension **.pevs**, and be located in the subfolder **..\Scripts\VFScripts** of the main PCIe Protocol Suite folder. Some other files might be included in the main script file using directive **%include**. (see the Teledyne LeCroy PCIe Protocol Suite File Based Decoding user manual for details).

The following schema presents a common structure of a verification script (this is similar to the content of the script template [**VSTemplate.pev_**] which is included with VSE):

```
#   VS1.pevs
#
#   Verification script
#
#   Brief Description:
#   Performs specific verification
#

##########################################################################################
   #  Module info
##########################################################################################
#  Filling of this block is necessary for proper verification script operation...
##########################################################################################
set ModuleType  = "Verification Script";            # Should be set for all verification scripts
set OutputType  = "VS";                             # Should be set for all verification scripts that
                                                    # output only Report string and Result.
set InputType   = "VS";

set DecoderDesc = "<Your Verification Script description>"; # Optional

##########################################################################################################
#
# include main Verification Script Engine definitions
#
%include  "VSTools.inc"                              # Should be set for all verification scripts


##########################################################################
#                           Global Variables and Constants
##########################################################################

# Define your verification script-specific global variables and constant in this section...
# (Optional)

    const MY_GLOBAL_CONSTANT = 10;
    set g_MyGlobalVariable   = 0;

##########################################################################


##########################################################################
#   OnStartScript()
##########################################################################
#
#        It is a main intialization routine for setting up all necessary
#        script parameters before running the script.
#
##########################################################################
```

```
OnStartScript()
{
  ##########################################################################################
  # Specify in the body of this function the initial values for global variables
  # and what kinds of trace events should be passed to the script.
  # ( By default, all packet level events from all channels
  #  are passed to the script.
  #
  #  For details – how to specify what kind of events should be passed to the script
  #  please see the topic 'sending functions'.
  #
  #  OPTIONAL.
  ##########################################################################################

   g_MyGlobalVariable   = 0;
    # Uncomment the line below - if you want to disable output from
    # ReportText()-functions.
    #
    # DisableOutput();
}
##########################################################################################
#   ProcessEvent()
##########################################################################################
#
##########################################################################################
# It is a main script function called by the application when the next waited event
# occured in the evaluated trace.
#
# !!! REQUIRED !!! – MUST BE IMPLEMENTED IN VERIFICATION SCRIPT
##########################################################################################

ProcessEvent()
{

    #  Write the body of this function depending upon your needs.
    #  It might require branching on event type:
    # select {
    #   in. TraceEvent == … : …
    #   in. TraceEvent == … : …
    #  …
    # }
    return Complete();
}

##########################################################################################
#   OnFinishScript()
##########################################################################################
#
##########################################################################################
# It is a main script function called by the application when the script completed
# running. Specify in this function some resetting procedures for a successive run
# of this script.
#
#  OPTIONAL.
##########################################################################################
OnFinishScript()
{
    return 0;
}

##########################################################################################

##########################################################################################
#   Additional script functions.
##########################################################################################
#
# Write your own script-specific functions here...
#
##########################################################################################
MyFunction( arg )
{
        if( arg == "My Arg" ) return 1;
```

```
        return 0;
}
```

3 Interaction between PCIe Protocol Suite and a verification script

When a user runs a script against a pre-recorded trace, the following sequence occurs:

Prior to sending information to the script's main processing function **ProcessEvent()**, VSE looks for the function **OnStartScript()** and calls it if it is found. In this function, setup actions are defined, such as specifying the kind of trace events that should be passed to the script and setting up initial values for script-specific global variables.

Next, the VSE parses the recorded trace to verify that the current packet or other event meets specific criteria – if it does, VSE calls the script's main processing function **ProcessEvent()**, placing information about the current event in the script's input context variables.

(Please refer to the topic **Input context variables** later in this document for a full description of verification script input context variables)

ProcessEvent() is the main verification routine for processing incoming trace events. This function must be present in all verification scripts. When the verification program consists of a few stages, the **ProcessEvent()** function processes the event sent to the script, verifies that information contained in the event is appropriate for the current stage, and decides if VSE should continue running the script or, if the whole result is clear on the current stage, tell VSE to complete execution of the script.

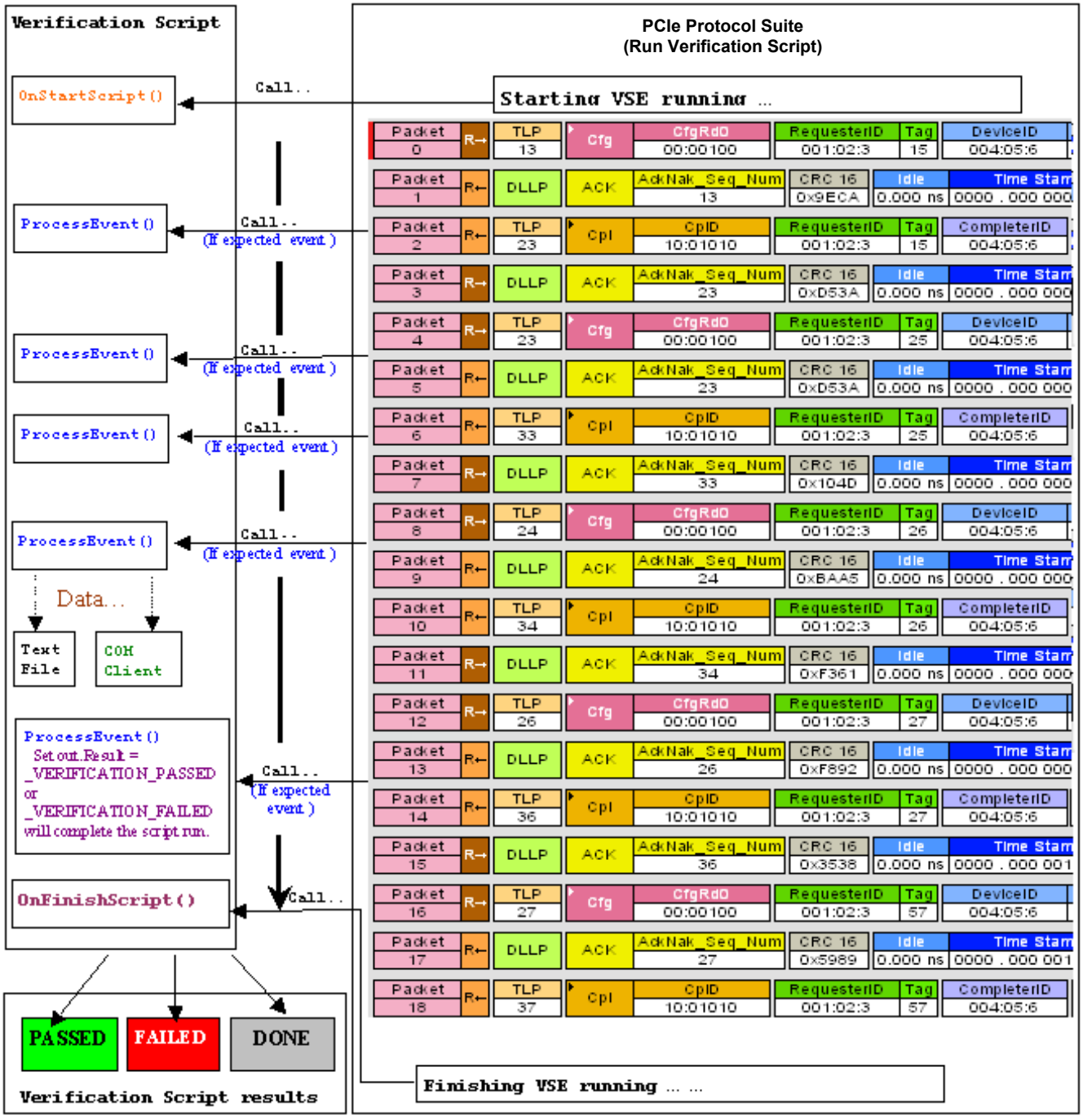
The completion of the test before the entire trace has been evaluated is usually done by setting the output context variable in this manner:
out.Result = _VERIFICATION_PASSED or **_VERIFICATION_FAILED**.

(Please refer to the topic **Output context variables** later in this document for a full description of verification script output context variables)

NOTE: Not only does a verification script evaluate recorded traces against some criteria, but it can also extract information of interest and post-process it later by some third-party applications. (There is a set of script functions allowing you to save extracted data in text files or send it to other applications, via COM/Automation interfaces.)

When the script has completed running, VSE looks for the function **OnFinishScript()** and calls it if found. In this function, some resetting procedures can be done.

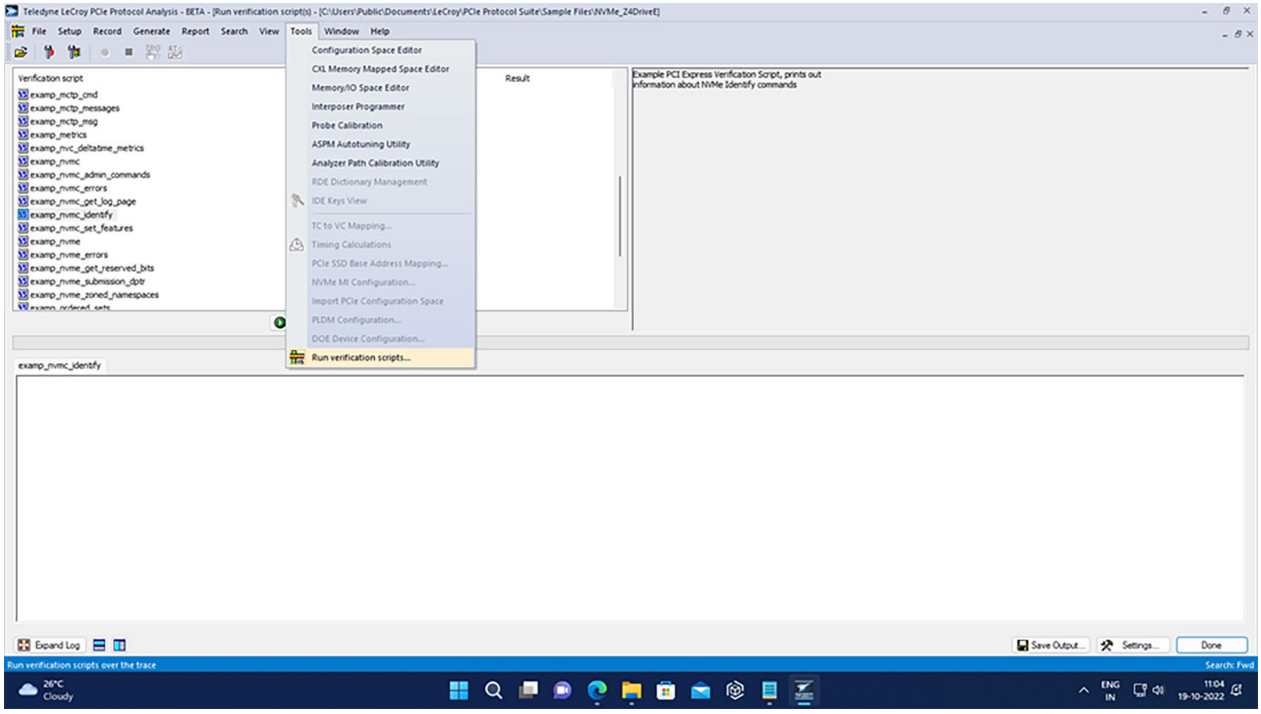
The following figure illustrates the interaction between the PCIe Protocol Suite™ application and a running verification script:



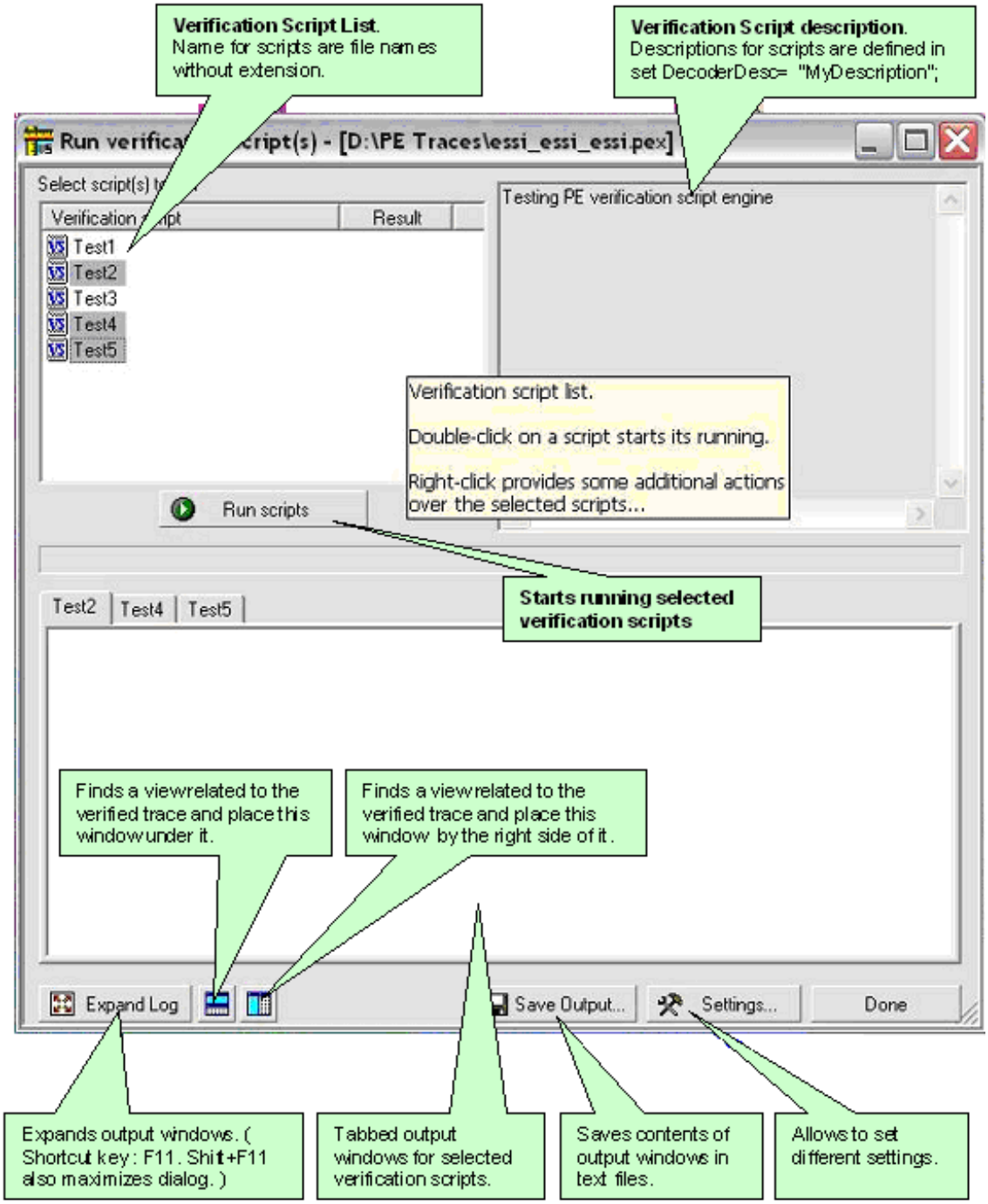
Note: The Verification script result "DONE" occurs when the script has been configured to extract and display some information about the trace, but not to display PASSED/FAILED results. To configure a script so that it only displays information – place a call somewhere in your script to the function **ScriptForDisplayOnly()in OnStartScript()**, for example.

4 Running verification scripts from the PCIe Protocol Suite

In order to run a verification script over a trace - open the PCIe Protocol Suite™, select **Tools\Run verification scripts,** or push the icon on the main toolbar (if it is not hidden).

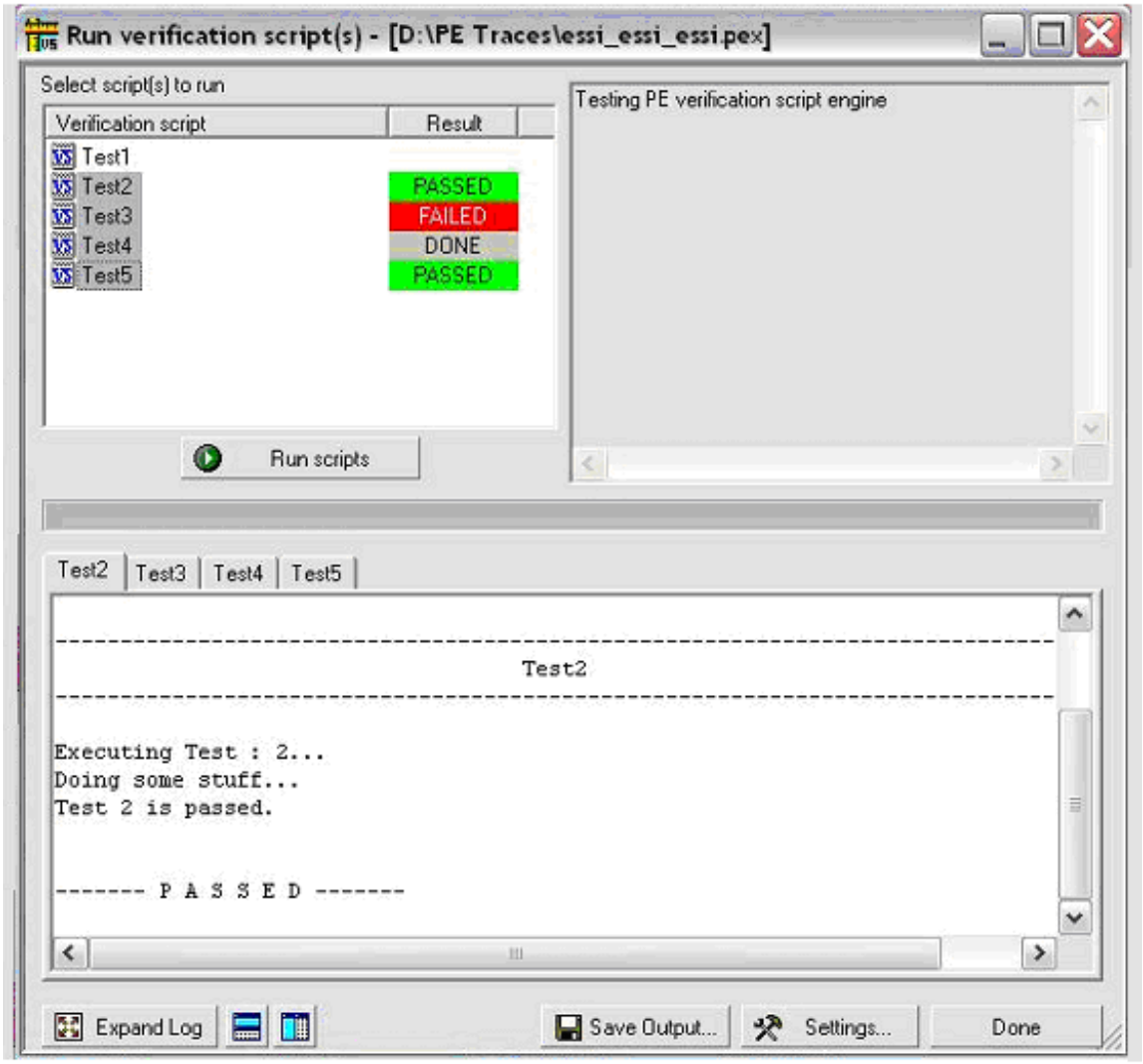


The special dialog opens displaying a list of verifications scripts. You can select one script to run, or several scripts from the list to run in parallel:

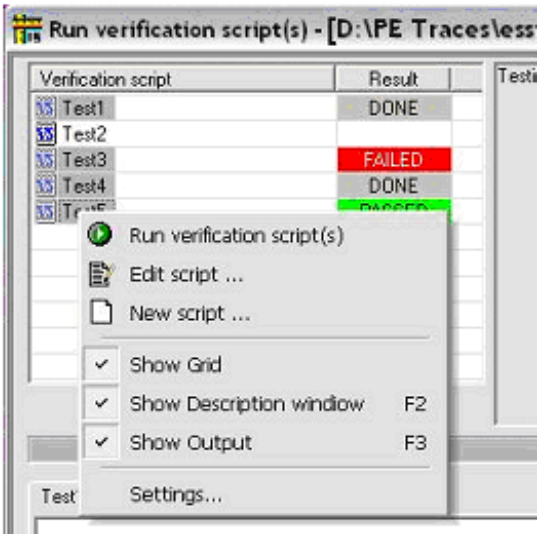


4.1 Running verification scripts

Push the button **Run scripts** after you selected the desired script(s) to run. VSE starts running the selected verification script(s), show script report information in the output windows, and present results of verifications in the script list:



Right-click in script list opens a pop-up menu with options for performing additional operations on the selected scripts:



Run verification script(s): Starts running selected script(s).

Edit script: Allows editing of the selected script(s) using whatever editor was specified in Editor settings.

New script: Creates a new script file using the template specified in Editor settings.

Show Grid: Shows/hides a grid in the verification script list**.**

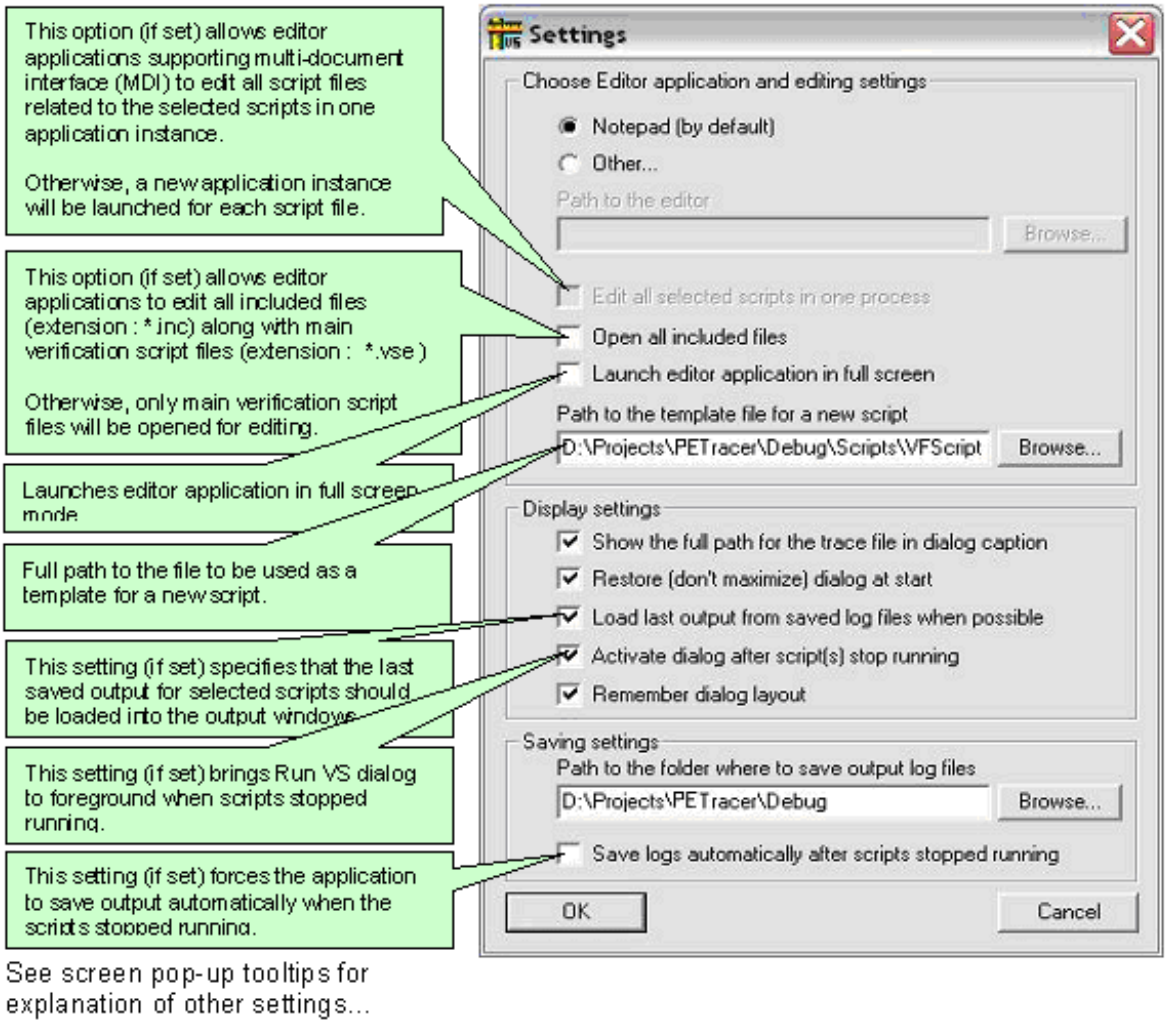
Show Description window: Shows/hides the script description window**. (Shortcut key : F2)**

Show Output: Shows/hides the script output windows. **(Shortcut key : F3)**

Settings: Opens a special Setting dialog which allows you to specify different settings for VSE.

4.2 VSE GUI Settings

After choosing **Settings**, the following dialog appears:



See screen pop-up tooltips for explanation of other settings...

5 Verification Script Engine Input Context Members

All verification scripts have input contexts – some special structures whose members are filled by the application and can be used inside of the scripts (for more details about input contexts – please refer to the *File-Based Decoding (FBD) Manual*). The verification script input contexts have two sets of members:

Trace event-independent set of members.
Trace event -dependent set of members.

5.1 Trace event-independent set of members

This set of members is defined and can be used for any event passed to script:

in.Level: Transaction level of the trace event (0 = packets, 1 = link transactions, 2 = split transactions, 3 = NVM transactions, 5 = AHCI transactions, 6 = ATA transactions, 9 = NVM commands, 11 = MCTP messages, 12 = MCTP commands)

in.Index: Index of the event in the trace file (frame number for frames, sequence number for sequences)

in.Time: Time of the event (type: list, having the format: 2 sec 125 ns -> [2 , 125]. (See 9.1 VSE Time Object for details)

in.Time2: Time of the event (type: list, having the format: 2 sec 125 ns 38ps -> [2 , 125 , 38]. (See 9.1 VSE Time Object for details)

in.Channel: Channel where the event occured. (may be **_CHANNEL_1** (1) or **_CHANNEL_2** (2) indicating which direction of the PE link the event occurred)

in.TraceEvent: Type of trace event (application predefined constants are used. See the list of possible events, below)

in.Notification: Type of notification (application predefined constants are used. Currently, no notifications are defined)

5.2 Trace event-dependent set of members

This set of members is defined and can be used only for a specific events or after calling some functions filling out some of the variables:

5.2.1 All packet/transaction-specific set of members

Members of this set are valid for any event.

in.Payload: Bit source of the frame/sequence payload (you can extract any necessary information using the **GetNBits(), NextNBits(),** or **PeekNBits()** functions.

Note: in.Payload does not support SCSI.

in.PayloadLength: Length (in bytes of the retrieved payload)

in.LinkWidth: Link Width recorded for this packet. Possible values 1, 2, 4, 8, 16, 32, UNDEFINED (is used only for Link Events, has value 255, and has a according constant name **_LINK_WIDTH_UNDEFINED**) represent the number of lanes on the link. Only available at the Packet and Link Transaction levels.

in.Speed: Speed of this packet or link transaction: 0 = 2.5GT/s, 1 = 5.0 GT/s, 2 = 8.0 GT/s, 3 = 16.0 GT/s, 4 = 32GT/s 5 = 64GT/s, 255 = UNDEFINED. The following constants are defined for the possible values **_SPEED_GEN1, _SPEED_GEN2**, **_SPEED_GEN3, _SPEED_GEN4**, **_SPEED_GEN5**, **_SPEED_GEN6**, **_SPEED_UNDEFINED** (**_SPEED_UNDEFINED** for Link Events only)**.**
Only available at the Packet and Link Transaction levels.

Error-related Variables (used for passing all the detected packet error types to the script)

in.HasErrors: Indicates the presence of any general error type in the current packet or critical packet-type-specific errors. It is a logical OR of **in.ErrorDisparity**, **in.ErrorSymbol, in.ErrorDelimiter, in.ErrorEndBad, in.ErrorAllignment, in.ErrorLength, in.ErrorWrongSymbol, in.BadLCRC** [TLP and DLLP Packet Types], **in.BadECRC** [TLP Packet Type], **in.MsgErrorG3LenCheck** [TLP Packet Type], **in.TsErrorDataRate**, **in.TsErrorFormat, in.TsErrorParity, in.TsErrorRsrvField** [TS1 and TS2 Packet Type] and **in.G3ErrorFraming** [DLLP Packet Type]. If this variable is 1, one or more of the errors indicated are present. If it is 0 (zero), the errors indicated are not present.

in.ErrorDisparity: If set to a non-zero value, indicates presence of Running Disparity error(s) in this packet.

in.ErrorSymbol: If set to a non-zero value, indicates presence of Symbol (10-bit Code) error(s) in this packet.

in.ErrorWrongSymbol: If set to a non-zero value, indicates a K symbol was received where a D symbol was expected, or vice versa.

in.ErrorDelimiter: If set to a non-zero value, indicates presence of Delimiter error(s) in this packet.

in.ErrorEndBad: If set to a non-zero value, indicates presence of an EDB symbol in this packet.

in.ErrorAlignment: If set to a non-zero value, indicates presence of Alignment error(s) in this packet.

in.ErrorLength: If set to a non-zero value, indicates presence of Bad Length error(s) in this TLP packet.

in.HasIdleErrors: Indicates presence of Idle errors in the current packet. If set, one of the following is set, indicating the presence of error(s) between this packet and the previous packet on this direction of the link:

in.IdleErrorDisparity: If set to a non-zero value, indicates presence of Running Disparity error(s).

in.IdleErrorSymbol: If set to a non-zero value, indicates presence of Symbol (10-bit Code) error(s).

in.IdleErrorSkip: If set to a non-zero value, indicates presence of Skip error(s).

in.IdleErrorData: If set to a non-zero value, indicates presence of Logical Idle data pattern error(s).

in.TsErrorDataRate: If set to a non-zero value, indicates presence of Training sequence data rate error(s).

in.TsErrorFormat: If set to a non-zero value, indicates presence of Training sequence format error(s).

in.TsErrorParity: If set to a non-zero value, indicates presence of Training sequence parity error(s).

in.TsErrorRsrvField: If set to a non-zero value, indicates presence of Training sequence reserved fields not zero error(s).

Note: For CRC error variables, see the specific packet type variable sets.

in.LtssmState: Only available at the Packet level. If set to a non-zero value, indicates the Ltssm state definition for this packet. The returned value is list in the following format: *[Ltssm state, Ltssm sub-state, Is state valid, Ltssm sub-sub-state]*.

in.LtssmStateIdle: Only available at the Packet level. To detect Idle states, use in pair with in.LtssmState. If set to a non-zero value, indicates the Ltssm Idle state definition for this packet. The returned value is list in the same format as for in.LtssmState.

The following constants are defined for the possible Ltssm state values:

Constant	PCIe LTSSM State/Sub-state
LTSSM_STATE_UNDEFINED,	State is not defined
LTSSM_STATE_DETECT,	Detect
LTSSM_STATE_POLLING,	Polling
LTSSM_STATE_CONFIG,	Configuration
LTSSM_STATE_L0,	L0
LTSSM_STATE_RECOVERY,	Recovery
LTSSM_STATE_HOT_RESET,	Hot Reset
LTSSM_STATE_DISABLED,	Disabled
LTSSM_STATE_LOOP_BACK,	LoopBack
LTSSM_STATE_L0S,	L0s
LTSSM_STATE_L1,	L1
LTSSM_STATE_L2,	L2
LTSSM_STATE_RECOVERY_RCVRLOCK,	Recovery.RcvrLock
LTSSM_STATE_RECOVERY_RCVRCFG,	Recovery.RcvrCfg
LTSSM_STATE_RECOVERY_RCVRIDL,	Recovery.Idle
LTSSM_STATE_RECOVERY_RCVREQL,	Recovery.Equalization
LTSSM_STATE_RECOVERY_RCVRSPD,	Recovery.Speed
LTSSM_STATE_RECOVERY_RCVREQL_1,	Recovery.Equalization Phase I
LTSSM_STATE_RECOVERY_RCVREQL_2,	Recovery.Equalization Phase II
LTSSM_STATE_RECOVERY_RCVREQL_3,	Recovery.Equalization Phase III
LTSSM_STATE_CFG_LINKWIDTH_STRT	Configuration.Linkwidth.Start
LTSSM_STATE_CFG_LINKWIDTH_ACPT	Configuration.Linkwidth.Accept
LTSSM_STATE_CFG_LANENUM_WAIT_ACPT,	Configuration.Lanenum.Wait / Accept
LTSSM_STATE_CFG_COMPLETE,	Configuration.Complete
LTSSM_STATE_CFG_IDLE,	Configuration.Idle
LTSSM_STATE_L1_1_OR_L1_2	L1.1 or L1.2
LTSSM_STATE_POLLING_ACTIVE	Polling.Active

LTSSM_STATE_POLLING_COMPLIANCE	Polling.Compliance
LTSSM_STATE_POLLING_CONFIGURATION	Polling.Configuration
LTSSM_STATE_POLLING_SPEED	Polling.Speed
LTSSM_STATE_LOOP_BACK_ENTRY	Loopback.Entry
LTSSM_STATE_LOOP_BACK_ACTIVE	Loopback.Active
LTSSM_STATE_LOOP_BACK_EXIT	Loopback.Exit
LTSSM_STATE_L0_L0P	L0p
LTSSM_STATE_RECOVERY_RCVREQL_0	Recovery.Equalization Phase 0
LTSSM_STATE_L0_L0P_UP	L0p Upsize
LTSSM_STATE_L0_L0P_DOWN	L0p Downsize

Note: Please refer to the **examp_ltssm.pevs** sample script for an example of how to process Ltssm states.

5.2.2 DLLP-specific set of members

Valid for data link layer packets only. Undefined for other events.

in.DLLPType: Contains the numeric encoding of the DLLP type. The following possible values are defined by VSE and the corresponding constants can be used by scripts (PCIe Flit/Non-Flit modes):

```
DLLP_TYPE_ACK                    = 0x0;
DLLP_TYPE_NAK                    = 0x1;
DLLP_TYPE_DATA_LINK_FEATURE      = 0x2;
DLLP_TYPE_INIT_FC1_P             = 0x4;
DLLP_TYPE_INIT_FC1_NP            = 0x5;
DLLP_TYPE_INIT_FC1_CPL           = 0x6;
DLLP_TYPE_INIT_FC2_P             = 0xC;
DLLP_TYPE_INIT_FC2_NP            = 0xD;
DLLP_TYPE_INIT_FC2_CPL           = 0xE;
DLLP_TYPE_UPDATE_FC_P            = 0x8;
DLLP_TYPE_UPDATE_FC_NP           = 0x9;
DLLP_TYPE_UPDATE_FC_CPL          = 0xA;
DLLP_TYPE_VENDOR                 = 0x3;

DLLP_TYPE_PM_ENTER_L1            = 0x10;
DLLP_TYPE_PM_ENTER_L23           = 0x11;
DLLP_TYPE_PM_ACT_STATE_REQUEST_L1 = 0x13;
DLLP_TYPE_PM_REQUEST_ACK         = 0x14;
DLLP_TYPE_NOP                    = 0x31;

DLLP_TYPE_LINK_MANAGEMENT        = 0x12; (PCIe Flit Mode)
DLLP_TYPE_NOP2                   = 0x15; (PCIe Flit Mode))

DLLP_TYPE_INVALID                = 0x7;
DLLP_TYPE_MR                     = 0xB;
```

in.AckNak_SeqNum: Field value (valid only for Ack and Nak DLLPs), indicating which TLPs are affected by the acknowledgement

in.VC_ID: Virtual Channel ID (valid only for InitFC and UpdateFC DLLPs) (PCIe Flit/Non-Flit modes)

in.HdrFC: Credit value for headers of the type indicated by the DLLP type (valid only for InitFC and UpdateFC DLLPs) (PCIe Flit/Non-Flit modes)

in.HdrScale: Contains the scaling factor for headers of the indicated typev(PCIe Flit/Non-Flit modes)

in.DataFC: Credit value for payload data of the type indicated by the DLLP type (valid only for InitFC and UpdateFC DLLPs) (PCIe Flit/Non-Flit modes)

in.DataScale: Contains the scaling factor for data payload of the indicated type (PCIe Flit/Non-Flit modes)

in.VendorSpecific: 3-byte vendor-defined value in a Vendor-specific DLLP (PCIe Flit/Non-Flit modes)

in.FeatureAck: Data Link Feature DLLP related, Feature Ack flag (PCIe Flit/Non-Flit modes)

in.FeatureSupport: Data Link Feature DLLP related, indicates the Data Link Features supported and/or attribute values for the transmitting Port (PCIe Flit/Non-Flit modes)

DLLP Error related Variables

in.InvalidEncoding: If set to a non-zero value, indicates an invalid DLLP encoding.

in.RsvdField: If set to a non-zero value, indicates a reserved field is non-zero or in use.

in.G3ErrorFraming: If set to a non-zero value, indicates Symbol 1 is incorrect, but Symbol 0 is correct. The value of the incorrect Symbol 1 is stored in **in.G3ErrorSym1Val**.
Note: This is a PCIE Gen 3 DLLP error.

in.G3ErrorSym1Val: The value of an incorrect Symbol 1 if **in.G3ErrorFraming** is set to a non-zero value.

in.FCError: If set to a non-zero value, indicates a Flow Control initialization protocol violation.

in.BadCRC: Set to 1 if the DLLP has bad 16-bit CRC and to 0 otherwise.

PCIe Flit-Mode specific fields

in.LinkManagementTypeStr: Applicable only for Link Management DLLP. Possible values:
- "L0p"

in.LinkManagementL0pPriorityStr: Applicable only for Link Management DLLP. Possible values:
- "Normal"
- "High"

in.LinkManagementL0pCmdStr: Applicable only for Link Management DLLP. Possible values:
- "Request"
- "Request Ack"
- "Request Nak"

in.LinkManagementL0pLinkWidthStr: Applicable only for Link Management DLLP. Possible values:
- "x1"
- "x2"
- "x4"
- "x8"
- "x16"

in.LinkManagementResponsePayloadStr: Applicable only for Link Management DLLP. Possible values:
- "x1"
- "x2"
- "x4"
- "x8"
- "x16"

in.InvalidEncoding: Invalid DLLP encoding.

in.ErrorRsvdFld: Reserved fields not NULL.

Un-decoded Frame

in.Frame: Contains the complete DLLP frame, i.e. from SDP till END.

5.2.3 TLP-specific set of members

Valid for TLPs only, undefined for other events.

All TLPs

in.TLPType: Contains the numeric encoding of the TLP type. The following possible values are defined by VSE and the corresponding constants can be used by scripts (PCIe Flit/Non-Flit modes):

```
TLP_TYPE_ID_INVALID    = 0;
TLP_TYPE_ID_MRD32      = 1;
TLP_TYPE_ID_MRDLK32    = 2;
TLP_TYPE_ID_MWR32      = 3;
TLP_TYPE_ID_MRD64      = 4;
TLP_TYPE_ID_MRDLK64    = 5;
TLP_TYPE_ID_MWR64      = 6;
TLP_TYPE_ID_IORD       = 7;
TLP_TYPE_ID_IOWR       = 8;
TLP_TYPE_ID_CFGRD_0    = 9;
TLP_TYPE_ID_CFGWR_0    = 10;
TLP_TYPE_ID_CFGRD_1    = 11;
TLP_TYPE_ID_CFGWR_1    = 12;
TLP_TYPE_ID_MSG        = 13;
TLP_TYPE_ID_MSGD       = 14;
TLP_TYPE_ID_MSGAS      = 15;
TLP_TYPE_ID_MSGASD     = 16;
TLP_TYPE_ID_CPL        = 17;
TLP_TYPE_ID_CPLD       = 18;
TLP_TYPE_ID_CPLLK      = 19;
TLP_TYPE_ID_CPLDLK     = 20;
TLP_TYPE_ID_FETCHADD32 = 21;
TLP_TYPE_ID_FETCHADD64 = 22;
TLP_TYPE_ID_SWAP32     = 23;
TLP_TYPE_ID_SWAP64     = 24;
TLP_TYPE_ID_CAS32      = 25;
TLP_TYPE_ID_CAS64      = 26;
TLP_TYPE_ID_DMWR32     = 27;
TLP_TYPE_ID_DMWR64     = 28;
TLP_TYPE_ID_UIOWRCPL   = 29;
TLP_TYPE_ID_UIORDCPL   = 30;
TLP_TYPE_ID_UIOMRD     = 31;
TLP_TYPE_ID_UIORDCPLD  = 32;
TLP_TYPE_ID_UIOMWR     = 33;
```

in.IsIdeTLP: If set to a non-zero value, indicates an IDE TLP.

Note: For a comprehensive and most up to date list of constants and codes please review file \Users\Public\Documents\LeCroy\PCIe Protocol Suite\Scripts\VFScripts\VS_constants.inc

TLP Error-related Variables

in.InvalidEncoding: If set to a non-zero value, indicates an invalid TLP encoding.

in.ErrorRsvdFld: If set to a non-zero value, indicates a reserved field is non-zero or in use.

in.ErrorPayload: If set to a non-zero value, indicates the TLP Payload does not match the Length field, so that the TD field value does not correspond with the observed size.

in.ErrorLengthField: If set to a non-zero value, indicates the Length field is invalid.

in.ErrorTCField: If set to a non-zero value, indicates the TC field is invalid.

in.ErrorAttrField: If set to a non-zero value, indicates the Attr field is invalid.

in.ErrorByteEnables: If set to a non-zero value, indicates the TLP violates the Byte Enable rules.

in.MemErrorAddrLength: If set to a non-zero value, indicates the Address/Length combination causes a Memory Space access to cross a 4-KB boundary.

in.MemErrorWrongType: If set to a non-zero value, indicates the wrong bit format is being used. For example, for addresses below 4 GB, Requesters must use 32 bit format.

in.CfgErrorRegister: If set to a non-zero value, indicates an invalid register field for Cfg. Must be DWORD aligned.

in.MsgErrorRouting: If set to a non-zero value, indicates invalid Msg or MsgD routing.

in.MsgErrorG3LenCheck: If set to a non-zero value, indicates a CRC-4 and/or Parity check failed on a Gen3 TLP length field (in framing, not the header). **Note**: This is a PCIE Gen 3 error.

in.BadLCRC: Set to 1 if the TLP has bad LCRC, to 0 otherwise

in.BadECRC: Set to 1 if the TLP has bad ECRC (when it should be present), to 0 otherwise

Field values for all TLP types:

in.Type: Type of TLP field value

in.Fmt: Format of TLP field value

in.PSN: Packet Sequence Number for this TLP as set by the Data Link Layer

in. RequesterId: Requester ID value (Bus, Device and Function Number fields combined) (PCIe Flit/Non-Flit modes)

in.Tag: Tag field value (PCIe Flit/Non-Flit modes)

in.TC: Traffic Class field value (PCIe Flit/Non-Flit modes)

in.Snoop: Snoop attribute bit value (PCIe Flit/Non-Flit modes)

in.Ordering: Ordering attribute bit value (PCIe Flit/Non-Flit modes)

in.IDBasedOrdering: ID Based Ordering attribute bit value (PCIe Flit/Non-Flit modes)

in.Attributes: Attributes field value [all three bits (Snoop, Ordering, and IDBasedOrdering)] (PCIe Flit/Non-Flit modes)

in.TH: TLP Processing Hints bit value

in.TD: TLP Digest bit value

in.EP: Poisoned TLP bit value

in.AT: Address Type field value (PCIe Flit/Non-Flit modes)

in.Length: Length field value (PCIe Flit/Non-Flit modes)

in.LCRC: LCRC value as set by the Data Link Layer

in.ECRC: ECRC value (optional)

in.IsMCTPPacket: Returns 1 if TLP carries MCTP packet, 0 otherwise

in.TLPPrefixesList: Returns TLP prefix (only one for Non-Flit mode and all as list for Flit mode (should be used in conjuction with **in.FM_TLPPrefixesCount** value))

Field values dependent upon TLP type:

in.FirstDwBe: Byte Enable bits for the first DW of the payload (all TLPs except Completions and Messages) (PCIe Flit/Non-Flit modes)

in.LastDwBe: Byte Enable bits for the last DW of the payload (all TLPs except Completions and Messages) (PCIe Flit/Non-Flit modes)

in.Address: 32-bit Address value for IO, Configuration, and Mem-32 requests (PCIe Flit/Non-Flit modes)

in.AddressLo: Low 32 bits of the Address for Mem-64 requests and Messages routed by address (PCIe Flit/Non-Flit modes)

in.AddressHi: High 32 bits of the Address for Mem-64 requests and Messages routed by address (PCIe Flit/Non-Flit modes)

in.DeviceId: Requester ID value (Bus, Device and Function Number fields combined) for Configuration requests and Messages routed by ID (PCIe Flit/Non-Flit modes)

in.Register: Register address (Register Number and Extended Register Number combined) for Configuration requests (PCIe Flit/Non-Flit modes)

For Completion TLPs only:

in.CompleterId: Completer ID value (Bus, Device and Function Number fields combined) (PCIe Flit/Non-Flit modes)

in.ComplStatus: Completion Status field value (PCIe Flit/Non-Flit modes)

in.BCM: Byte Count Modified bit value

in.ByteCount: Remaining Byte Count field value (PCIe Flit/Non-Flit modes)

in.LowerAddr: Lower Address for starting byte of completion field value (PCIe Flit/Non-Flit modes)

For Message TLPs only:

in.MessageCode: Message Code field value (PCIe Flit/Non-Flit modes)

in.MessageRoute: Message Routing field value (from the TLP Type field) (PCIe Flit/Non-Flit modes)

Note: For a comprehensive and most up to date list of constants and codes please review file \Users\Public\Documents\LeCroy\PCIe Protocol Suite\Scripts\VFScripts\VS_constants.inc

For Memory TLPs only:

in.NW: No Write flag in Translation Requests (Non-Flit mode), in OHC-A1 (Flit mode)

For Configuration Write Requests and Read Completions:

in.RegisterData: 32-bit value written to or read from a configuration register (for convenience of processing the configuration requests, as it also can be obtained from the Payload)

Un-decoded Frame

in.Frame: Contains the complete TLP frame, i.e. STP till END.

For IDE TLPs only (PCIe Flit/Non-Flit modes, unless specified otherwise):

in.IdeEncryptedPayload: Ide encrypted payload binary data.

in.IdeEncryptedPayloadLength: Ide encrypted payload binary data length in bytes.

in.IdeDecryptedPayload: Ide decrypted payload binary data.

in.IdeDecryptedPayloadLength: Ide decrypted payload binary data length (in bytes).

in.IdeStreamID: IDE Stream ID

in.IdeSubStream: IDE Sub Stream

in.IdeM: IDE MAC present, PCIe Non-Flit mode only

in.IdeP: IDE PCRC present, PCIe Non-Flit mode only

in.IdeK: IDE K bit from prefix/OHC-C

in.IdeT: IDE T bit from prefix/OHC-C

in.IdePartialHeaderEncryptionMode: IDE Partial Header Encryption Mode derived from IDE Extended Capability

in.IdeDecryptedFirstDwBe: Decrypted First DW Byte Enables if partial header encryption took place, and authentication was successful. Invalid when “in. FirstDwBe” is invalid

in.IdeEncryptedFirstDwBe: Encrypted First DW Byte Enables if partial header encryption took place. Invalid when “in. FirstDwBe” is invalid

in.IdeDecryptedLastDwBe: Decrypted Last DW Byte Enables if partial header encryption took place, and authentication was successful. Invalid when “in. LastDwBe” is invalid

in.IdeEncryptedLastDwBe: Encrypted Last DW Byte Enables if partial header encryption took place. Invalid when “in. LastDwBe” is invalid

in.IdeDecryptedAddress: Decrypted 32-bit address if partial header encryption took place, and

authentication was successful. Invalid when “in.Address” is invalid

in.IdeEncryptedAddress: Encrypted 32-bit address if partial header encryption took place. Invalid when “in.Address” is invalid

in.IdeDecryptedAddressLo: Low 32 bits of decrypted 64-bit address if partial header encryption took place, and authentication was successful. Invalid when “in. AddressLo” is invalid

in.IdeEncryptedAddressLo: Low 32 bits of encrypted 64-bit address if partial header encryption took place. Invalid when “in. AddressLo” is invalid

in.IdeDecryptedAddressHi: High 32 bits of decrypted 64-bit address if partial header encryption took place, and authentication was successful. Invalid when “in. AddressHi” is invalid

in.IdeEncryptedAddressHi: High 32 bits of encrypted 64-bit address if partial header encryption took place. Invalid when “in. AddressHi” is invalid

in.HasIdeErrors: returns 1 if the TLP has any errors related to IDE

in.IdeDecryptionError: decryption error text description

in.IDEErrorIntegrity: returns 1 if the error is present, 0 otherwise

in.IDEErrorAggregationWithoutMAC: returns 1 if the error is present, 0 otherwise

in.IDEErrorMissingKeyInfo: returns 1 if the error is present, 0 otherwise

in.IDEErrorPCRC: returns 1 if the error is present, 0 otherwise

in.IDEErrorHasECRC: returns 1 if the error is present, 0 otherwise. PCIe Non-Flit mode only

in.IDEErrorPBit: returns 1 if the error is present, 0 otherwise. PCIe Non-Flit mode only

in.IDEErrorMissingPCRC: returns 1 if the error is present, 0 otherwise. PCIe Non-Flit mode only

in.IDEErrorMissingMAC: returns 1 if the error is present, 0 otherwise. PCIe Non-Flit mode only

in.IDEErrorNotPermittedType: returns 1 if the error is present, 0 otherwise

in.IDEErrorPrefixOrder: returns 1 if the error is present, 0 otherwise. PCIe Non-Flit mode only

in.IDEErrorIncorrectSubstream: returns 1 if the error is present, 0 otherwise

in.IDEErrorRsvrSubstream: returns 1 if the error is present, 0 otherwise

in.IDEErrorFMIncorrectTS: returns 1 if the error is present, 0 otherwise. PCIe Flit mode only

in.IdeMacData: IDE MAC binary data

in.IdeMacSize: IDE MAC data size in bytes

in.IdeEncryptedPCRC: encrypted PCRC value

in.IdeDecryptedPCRC: decrypted PCRC value, if authentication was successful

in.IdeCalculatedPCRC: calculated PCRC value, if authentication was successful

in.IdeKeyData: decryption key binary data

in.IdeKeySize: decryption key size in bytes

in.IdeIVData: current aggregation IV binary data

in.IdeIVSize: current aggregation IV size in bytes

For TLPs in Flit mode only:

in.FM_TLP_OHC: OHC full value

in.FM_TLP_TS: TS value

in.FM_TLP_OHC_A: OHC-A is present

in.FM_TLP_OHC_B: OHC-B is present

in.FM_TLP_OHC_C: OHC-C is present

in.FM_TLP_OHC_E: OHC-E is present

in.FM_TLP_OHC_A_1_4_PV: OHC-A1, OHC-A4 PV value

in.FM_TLP_OHC_A_1_4_PASID: OHC-A1, OHC-A4 PASID value

in.FM_TLP_OHC_A_1_PMR: OHC-A1 PMR value

in.FM_TLP_OHC_A_1_ER: OHC-A1 ER value

in.FM_TLP_OHC_A_DSV: OHC-A DSV value

in.FM_TLP_OHC_A_DS: OHC-A Destination Segment value

in.FM_TLP_OHC_A_CS: OHC-A5 Completer Segment value

in.FM_TLP_OHC_B_AV: OHC-B AV value

in.FM_TLP_OHC_B_AMA: OHC-B AMA value

in.FM_TLP_OHC_B_ST: OHC-B ST value

in.FM_TLP_OHC_B_HV: OHC-B HV value

in.FM_TLP_OHC_B_PH: OHC-B PH value

in.FM_TLPPrefixesCount: count of TLP prefixes

in.FM_TLPPrefixTypesStrList: names of the TLP prefixes, could be present all of them at the same time, should be used in conjuction with **in.FM_TLPPrefixesCount** value. Possible values:
- "FlitModePrefix"
- "VendPrefixL0"
- "VendPrefixL1"

in.FM_TLPPrefixLocalDedicatedCredits: TLP Uses Dedicated Credits flag

5.2.4 PCIe Flit specific set of members

Valid for PCIe Flit events only, undefined for other events.

in.PCIeFlitType: Contains the numeric encoding of the PCIe Flit type. The following possible values are defined by VSE and the corresponding constants can be used by scripts:

```
PCIE_FLIT_TYPE_IDLE      = 0
PCIE_FLIT_TYPE_NOP       = 1
PCIE_FLIT_TYPE_PAYLOAD   = 2
```

in.PCIeFlitSeqNumber: 10-bit sequence number applied to Flits.
in.PCIeFlitIsReplay: Is current flit a replay flit.
in.PriorFlitTypeStr: Prior Flit type, possible values:
- "IDLE/NOP"
- "Payload"

5.2.4.1 PCIe Flit DLP specific set of members

in.ReplayCmdStr: name of Replay command, possible values:
- "Explicit"
- "Ack"
- "Std Nak"
- "Sel Nak"

in.DlpHeaderSeqNum: 10-bit sequence number from DLP part

in.TypeOfDlpPayloadStr: Type of DLLP payload, possible values:
- "Dllp"
- "OptimizedUFC"
- "FlitMarker"

For Optimized Update FC DLLPs payload type only:

in.OptimizedUFC_NPHdrFC: Shared Non-Posted HdrFC – Shared credits on this VC

in.OptimizedUFC_PHdrFC: Shared Posted HdrFC – Shared credits on this VC

in.OptimizedUFC_PDataFC: Shared Posted DataFC – Shared credits on this VC

in.VC_ID: Virtual channel, see **in.VC_ID** in the DLLP-specific set of members

For Flit Marker DLLPs payload type only:

in.FlitStatusStr: Indicates the validity of the Last TLP in this Flit, possible values:
- "No Special Info"
- "Last TLP Nullified"
- "Last TLP Poisoned"

in.PTM_Message: This bit indicates that this Flit contains the last symbol of a non-nullified PTM Message

5.2.5 SMBus-specific set of members

Valid for SMBus events only, undefined for other events. Please refer to the **examp_smbus.pevs** sample script for an example of how to process SMBus events.

All SMBus events

in.IsARPPacket: Equals to 1 if this packet is SMBus ARP one, 0 for other packet types.

in.SMBusPacket: Returns SMBus packet in raw format, i. e. you can access any bytes directly. Packet size is the sizeof(in.SMBusPacket).

in.SlaveAddress: The 7-bit slave address that master wants to address on the bus.

in.Wr: The direction of the data transfer (R/W#); a ZERO indicates a transmission (WRITE) while a ONE indicates a request for data (READ).

Common commands for SMBus ARP and MCTP

in.ByteCount : Byte Count field for MCTP and "Get UDID", "Assign Address" SMBus ARP commands.

in.CalculatedByteCount : calculated Byte Count field for MCTP and "Get UDID", "Assign Address" SMBus ARP commands.

in.CalculatedPEC : calculated Packet Error Code.

in.PEC : transmitted Packet Error Code.

MCTP messages

in.CommandCode: SMBus Command Code.

in.DestSlaveAddress: The slave address of the target device for the local SMBus link. The same as in.SlaveAddress.

in.MCTPFlagOne: This bit shall be set to 1b. The value enables MCTP to be differentiated from IPMI over SMBus and IPMB (IPMI over I2C) protocols.**in.SourceSlaveAddress:** The slave address of the source device.

SMBus ARP Commands

in.ARPCommandCode: Current SMBus ARP command code :

Constant	Value	Meaning
_SMBUS_ARP_COMMAND_CUSTOM	0	Incorrect or unsupported command
_SMBUS_ARP_COMMAND_RESERVED	1	Reserved command code
_SMBUS_ARP_COMMAND_PREPARE_TO_ARP	2	"Prepare to ARP" command
_SMBUS_ARP_COMMAND_RESET_DEVICE_GENERAL	3	"Reset device (directed)" command
_SMBUS_ARP_COMMAND_RESET_DEVICE_DIRECTED	4	"Reset device (directed)" command
_SMBUS_ARP_COMMAND_GET_UDID_GENERAL	5	"Get UDID (general)" command

_SMBUS_ARP_COMMAND_GET_UDID_DIRECTED	6	"Get UDID (directed)" command
_SMBUS_ARP_COMMAND_ASSIGN_ADDRESS	7	"Assign address" command
_SMBUS_ARP_COMMAND_NOTIFY_ARP_MASTER	8	"Notify ARP master" command

in.UDID : raw UDID data buffer for commands with UDID. Can be accessed like UDID[0], UDID[1] etc. UDID[0] is Device Capabilities field, UDID[1] is Version / Revision field etc.

in.UDID_DeviceCapabilities : Device Capabilities field from UDID.

in.UDID_AddressType : Device Capabilities's Address Type field from UDID :

Constant	Value	Meaning
_SMBUS_ARP_SMBUS_ADDRESS_TYPE_FIXED	00b	Fixed Address device
_SMBUS_ARP_SMBUS_ADDRESS_TYPE_DYNAMIC_PERSISTENT	01b	Dynamic and persistent address device
_SMBUS_ARP_SMBUS_ADDRESS_TYPE_DYNAMIC_VOLATILE	10b	Dynamic and volatile address device
_SMBUS_ARP_SMBUS_ADDRESS_TYPE_RANDOM	11b	Random number device

in.UDID_PECSupported : Device Capabilitites's PEC Supported field.

in.UDID_VersionRevision : Version/Revision field from UDID.

in.UDID_SMBusVer : SMBus Version field from UDID. All values that are not listed below are reserved :

Constant	Value	Meaning
_SMBUS_ARP_SMBUS_VERSION_1_0	0000b	SMBus 1.0
_SMBUS_ARP_SMBUS_VERSION_1_1	0001b	SMBus 1.1
_SMBUS_ARP_SMBUS_VERSION_2_0	0100b	SMBus 2.0
_SMBUS_ARP_SMBUS_VERSION_3_0	0101b	SMBus 3.0

in.UDID_UDIDVersion : Version/Revision's UDID Version field from UDID :

Constant	Value	Meaning
_SMBUS_ARP_SMBUS_UDID_VER_1	001b	UDID version 1 (defined for SMBus 2.0 release)

in.UDID_SiliconRevisionID : Version/Revision's Silicon Revision ID field from UDID.

in.UDID_VendorID : Vendor ID field from UDID.

in.UDID_DeviceID : Device ID field from UDID.

in.UDID_Interface : Interface field from UDID.

in.UDID_ZONE : Interface's ZONE field from UDID.

in.UDID_IPMI : Interface's IPMI field from UDID.

in.UDID_ASF : Interface's ASF field from UDID.

in.UDID_OEM : Interface's OEM field from UDID.

in.UDID_SubsystemVendorID : Subsystem Vendor ID field from UDID.

in.UDID_SubsystemDeviceID : Subsystem Device ID field from UDID.

in.UDID_VendorSpecificID : Vendor Specific ID field from UDID.

in.AssignedAddress : Assigned address field form Assign Address ARP command.

in.TargetedSlaveAddress : Targeted Slave Address field from Get UDID (directed) or Reset device ARP (directed) ARP commands.

in.DeviceAddress : Device Address field from Notify ARP master command.

in.DeviceSlaveAddress : Device Slave Address field from Get UDID (directed) ARP command.

in.Data : Data field (Data Byte High and Data Byte Low) field from Notify ARP master command.

5.2.6 MCTP messages specific set of members:

Valid for MCTP messages over VDM and SMBus transport events.

in.MCTP_HeaderVersion: Returns Header Version

in.MCTP_DestinationEId: Returns Destination Endpoint ID

in.MCTP_SourceEId: Returns Source Endpoint ID

in.MCTP_SOM: Returns Start Of Message bit

in.MCTP_EOM: Returns End Of Message bit

in.MCTP_PacketSequenceNumber: Returns Packet Sequence Number

in.MCTP_TagOwner: Returns Tag Owner value

in.MCTP_MsgTag: Returns Message Tag value

Note: Please refer to the **examp_mctp.pevs** sample script for an example of how to process MCTP packets.

5.2.7 Ordered Set specific set of members

in.OrderedSetType: Contains the numeric encoding of the Ordered Set type. The following possible values are defined by VSE and the corresponding constants can be used by scripts:

```
ORDSET_TYPE_TS0           = 0x21;
ORDSET_TYPE_TS1           = 0x02;
ORDSET_TYPE_TS2           = 0x03;
ORDSET_TYPE_FTS           = 0x04;
ORDSET_TYPE_EIOS          = 0x05;
ORDSET_TYPE_SKIP          = 0x06;
ORDSET_TYPE_PATN          = 0x07;
ORDSET_TYPE_EIEOS         = 0x08;
ORDSET_TYPE_SDS           = 0x0C;
ORDSET_TYPE_PATN_MODIFIED = 0x16;
ORDSET_TYPE_TS1_MOD       = 0x1C;
ORDSET_TYPE_TS2_MOD       = 0x1D;
```

Note: For a comprehensive and most up to date list of constants and codes please review file \Users\Public\Documents\LeCroy\PCIe Protocol Suite\Scripts\VFScripts\VS_constants.inc

in.EIEOS_RawSymbolsList: Contains the list of all symbols for all lanes

For Training Sequences (TS1, TS2 and Modified TS1, TS2), the following variables of the list type exist in the input context (the lists are arrays of integers with dimensions equal to the Link Width for the Training Sequence packet):

in.TS_LinkNumberList: Contains the Link Number parameter values for all lanes

in.TS_LaneNumberList: Contains the Lane Number parameter values for all lanes

in.TS_N_FTSList: Contains the N_FTS parameter values for all lanes

in.TS_TrainingControlList: Contains the Training Control bitmap parameter values for all lanes

in.TS_RawSymbolsList: Contains the list of all symbols for all lanes

in.TS_DataRateList: Contains the Data Rate parameter values for all lanes

The following parameters are only valid for Gen 3, Gen 4, Gen 5 TS1 and TS2:

in.TS_PreCursorList: Contains the Pre-Cursor parameter values for all lanes

in.TS_CursorList: Contains the Cursor parameter values for all lanes

in.TS_PostCursorList: Contains the Post-Cursor parameter values for all lanes

in.TS_EqControlList: Contains the Equalizer Control parameter values for all lanes

Next parameters are only valid for Modified TS1 and TS2:

in.ModTS_Usage: Contains the Modified TS Usage parameter values for all lanes

in.ModTS_Usage_Common: Contains the Modified TS Usage parameter common value from all lanes

in.ModTS_Info1: Contains the Modified TS Information 1 parameter values for all lanes

in.ModTS_TrainingSetMessageVID: Contains the Training Set Message Vendor ID parameter values for all lanes when Modified TS Usage is equal to 001b

in.ModTS_AlternateProtocolVID: Contains the Alternate Protocol Vendor ID parameter values for all lanes when Modified TS Usage is equal to 010b

in.ModTS_Info2: Contains the Modified TS Information 2 parameter values for all lanes

in.ModTS_ProtocolEnable_Common: Contains the list with 4 values { PCIe enable, CXL.io enable, CXL.cache enable, CXL.mem enable } which are common for symbol 12 in all lanes

Note: For Link Number and Lane Number values the special value of 0x1FF is used to indicate the PAD symbol. For Training Set Message Vendor ID and Alternate Protocol Vendor ID values the special value of 0x1FFFF is used to indicate that value is not applicable. Please refer to the **examp_ordered_sets.pevs** sample script for an example of how to process ordered Sets and Training Sequences, in particular.

5.2.8 Link Condition specific set of members

in.LinkConditionType: Contains the numeric encoding of the Link Condition type. The following possible values are defined by VSE and the corresponding constants can be used by scripts:

```
LINK_CONDITION_UNKNOWN          =0;    - Link Condition unknown.
LINK_CONDITION_LINK_DOWN        =1;    - “Link Up” Link Condition event
LINK_CONDITION_LINK_UP          =2;    - “Link Down” Link Condition event
LINK_CONDITION_SKEW             =16;   - “Deskewing” Link Condition event
LINK_CONDITION_LINK_WAKE_UP     =8;    - "Link Wake Up" Link Condition event
LINK_CONDITION_LINK_WAKE_DOWN   =64;   - "Link Wake Down" Link Condition event
LINK_CONDITION_LINK_PERST_UP    =4;    - "Link Perst Up" Link Condition event
LINK_CONDITION_LINK_PERST_DOWN  =32;   - "Link Perst Down" Link Condition event
LINK_CONDITION_LINK_CLKREQ_UP   =128;  - "Link Clock Request Up" Link Condition event
```

LINK_CONDITION_LINK_CLKREQ_DOWN =256; - "Link Clock Request Down" Link Condition event

Note: For a comprehensive and most up to date list of constants and codes please review file \Users\Public\Documents\LeCroy\PCIe Protocol Suite\Scripts\VFScripts\VS_constants.inc

5.2.9 L0p Link Condition specific set of members

in.LinkConditionTypeL0p: Contains the numeric encoding of the Link Condition type. The following possible values are defined by VSE and the corresponding constants can be used by scripts:

LINK_CONDITION_L0P_UPSIZING = 1; - “Upsizing” L0p Link Condition event
LINK_CONDITION_L0P_DOWNSIZING = 2; - “Downsizing” L0p Link Condition event
LINK_CONDITION_L0P_UPSIZE_COMPLETED = 3; - “Upsize Completed” L0p Link Condition event
LINK_CONDITION_L0P_DOWNSIZE_COMPLETED = 4; - “Downsize Completed” L0p Link Condition event
LINK_CONDITION_L0P_UPSIZE_FAILED = 5; - “Upsize Failed” L0p Link Condition event
LINK_CONDITION_L0P_SUDDEN_EXIT = 6; - “Sudden Exit” L0p Link Condition event

Note: For a comprehensive and most up to date list of constants and codes please review file \Users\Public\Documents\LeCroy\PCIe Protocol Suite\Scripts\VFScripts\VS_constants.inc

5.2.10 Link transaction-specific set of members

Valid for Link transactions only, undefined for other events.

All the TLP-specific values are present in the input context for Link transactions, depending upon the type of TLP for this Link transaction. In addition to that, the following value exists:

in.TransactionStatus: Status for this Link transaction. Can be one of three values: Implicitly Acknowledged, Explicitly Acknowledged, or Incomplete (Link Layer error). See file **VS_constants.inc** for encodings.

Metric values

The following values are defined in input context for Link Transactions that are related to Unit Metrics. To learn more about Unit Metrics, please refer to PCIe Protocol Suite™ Help.

in.Metric_NumOfPackets: Metric presenting the total number of packets that compose this Link Transaction, an integer value

in.Metric_ResponseTime: Metric presenting time it took to transmit this Link Transaction on the PE link, from the beginning of the first packet in the transaction to the end of the last packet in the transaction, a VSE time object value (see 9.1 VSE Time Object for details)

in.Metric_Throughput: Metric presenting transaction payload divided by response time, expressed in **kilobytes** per second, an integer value

in.Metric_PayloadBytes: Metric presenting number of data payload bytes this Link Transaction transferred, an integer value

Notes: For the incomplete Link Transactions only, the NumOfPackets metric is valid. In case of an incomplete Link Transaction, the ResponseTime metric value is set to `null`.

5.2.11 Split transaction-specific set of members

Valid for Split transactions only. Undefined for other events.

All the TLP-specific values **for the request TLP of the split transaction** are present in the input context for Link transactions, depending upon the type of TLP for this Link transaction. Also the common PayloadLength and Payload values reflect the total combined payload for the Split transaction. In addition to that, the following values exist:

in.CompletionStatus: Completion Status for this Split transaction. From the last completion of the response.

Metric values

The following values are defined in input context for Split Transactions that are related to Unit Metrics. To learn more about Unit Metrics please refer to PCIe Protocol Suite Help.

in.Metric_NumOfPackets: Metric presenting the total number of packets that compose this Link Transaction, an integer value

in.Metric_ResponseTime: Metric presenting time it took to transmit this Split Transaction on the PE link, from the beginning of the first packet in the transaction to the end of the last packet in the transaction, a VSE time object value (see 9.1 VSE Time Object for details)

in.Metric_LatencyTime: Metric presenting time measured from the end of the request transaction to the first completion transmitted in response to the request within this Split Transaction, a VSE time object value (see 9.1 VSE Time Object for details)

in.Metric_Throughput: Metric presenting transaction payload divided by response time, expressed in **kilobytes** per second, an integer value

in.Metric_PayloadBytes: Metric presenting number of data payload bytes this Split Transaction transferred, an integer value.

Notes: For the incomplete Link Transactions only, the NumOfPackets metric is valid. In case of an incomplete Link Transaction the ResponseTime metric value is set to `null`.

5.2.12 NVM transaction-specific set of members

Valid for NVM transactions only. Undefined for other events.

in.nvmeType: Returns NVMe register type. The value of 'in.nvmeType' depends on transaction event type and can be compared against the predefined values. The following possible values are defined by VSE and the corresponding constants can be used by scripts:

For _NVME_CONTROLLER_REG transaction event:

Constant	Value
_NVME_CAP	0
_NVME_VS	1
_NVME_INTMS	2
_NVME_INTMC	3
_NVME_CC	4
_NVME_RESERVED1	5
_NVME_CSTS	6
_NVME_NSSR	7
_NVME_AQA	8
_NVME_ASQ	9
_NVME_ACQ	10
_NVME_CMBLOC	11
_NVME_CMBSZ	12
_NVME_BPINFO	13
_NVME_BPRSEL	14
_NVME_BPMBL	15
_NVME_CMBMSC	16
_NVME_CMBSTS	17
_NVME_CMBEBS	18
_NVME_CMBSWTP	19
_NVME_RESERVED2	20
_NVME_PMRCAP	21
_NVME_PMRCTL	22
_NVME_PMRSTS	23
_NVME_PMREBS	24
_NVME_PMRSWTP	25
_NVME_PMRMSC	26
_NVME_RESERVED_CMD_SET_SPECIFIC	27

For _NVME_DOORBELL_REG transaction event:

Constant	Value
_NVME_ADMIN_SQTDBL	29
_NVME_ADMIN_CQHDBL	30
_NVME_SQYTDBL	31
_NVME_CQYHDBL	32

For _NVME_ADMIN_SUBMISSION_CMD transaction event:

Constant	Value
_NVME_ADMIN_SUBMISSION_Q_ENTRY	33

For _NVME_ADMIN_COMPLETION_CMD transaction event:

Constant	Value
_NVME_ADMIN_COMPLETION_Q_ENTRY	34

For _NVME_NVM_SUBMISSION_CMD transaction event:

Constant	Value
_NVME_IO_SUBMISSION_Q_ENTRY	35

For _NVME_NVM_COMPLETION_CMD transaction event:

Constant	Value
_NVME_IO_COMPLETION_Q_ENTRY	36

For _NVME_PRP transaction event:

Constant	Value
_CMD_PRP	151
_CMD_PRP_LIST	152

For _NVME_SGL transaction event:

Constant	Value
_SGL_DESCRIPTOR	154
_SGL_BIT_BUCKET	155
_MSGLP	156

For _NVME_TRANSFERED_DATA transaction event:

Constant	Value
_NVME_DATA	158

For _NVME_IDX_DAT_REG transaction event:

Constant	Value
_NVME_IDX	159
_NVME_DAT	160

For _NVME_INTERRUPT_REG transaction event:

Constant	Value

_NVME_INTERRUPT	161

in. nvmeQID: Defined for _NVME_DOORBELL_REG transaction event. Returns zero for Admin doorbells, or queue ID otherwise.

in. nvmeIndex: Defined for _NVME_DOORBELL_REG transaction event. Returns SQT for submission doorbells, and CQH – for completions.

in.nvmeRegisterValue: Returns the whole register value.
(Note: If you want to find a specific bit value in the register use bitwise operations)

in.nvmeTraHasError: If set to a non-zero value, indicates NVME transaction has errors.

In.nvmeErrorId: Contains the numeric encoding of the NVME error type. The following possible values are defined by VSE and the corresponding constants can be used by scripts:

Error type	Value	Error name	Error description
_NVME_ERROR_ACCESS_DIR_VIOLATION	1	Read-only registry write	Access direction violation
_NVME_ERROR_RESERVED_NOT_NULL	2	Reserved field is not zero	Reserved field is not zero
_NVME_ERROR_INVALID_FIELD_VALUE	4	Field value is not from specified set	Field value is not listed as a valid value
_NVME_ERROR_INCOMPLETE_TRA	16	Incomplete transaction	Incomplete transaction: size doesn't match expected
_NVME_ERROR_INCOMPLETE_SUB_TRA	32	Incomplete sub-transaction	Incomplete sub-transaction
_NVME_ERROR_ERROR_IN_SUB_TRA	64	Error in sub-transaction	Error in sub-transaction
_NVME_ERROR_LOGICAL_ERROR	128	Logical error	Logical error
_NVME_ERROR_QUEUE_ERROR	512	Queue Error	Queue Error
_NVME_ERROR_NO_ERROR	0	No Error	Correct transaction

in.nvmeErrorIdAsString: Contains an NVME error name

5.2.12.1 NVM transaction members specific to _NVME_ADMIN_SUBMISSION_CMD and to NVME_NVM_SUBMISSION_CMD events:

in.nvmeCID: Returns Command Id.

in.nvmePSDT: Returns whether PRPs or SGLs are used for any data transfer associated with the command. If cleared to '0', the command uses PRPs.

in.nvmeFUSE: In a fused operation, returns whether complex command is created by "fusing" together two simpler commands.

in. nvmeOpcode: Returns the NVMe command code. The value of 'in.nvmeOpcode' depends on transaction event type and can be compared against the predefined values. The following possible values are defined by VSE and the corresponding constants can be used by scripts:

For _NVME_ADMIN_SUBMISSION_CMD transaction event:

Constant	Value

_NVME_ASC_DELETE_IO_SQ	00h
_NVME_ASC_CREATE_IO_SQ	01h
_NVME_ASC_GET_LOG_PAGE	02h
_NVME_ASC_DELETE_IO_CQ	04h
_NVME_ASC_CREATE_IO_CQ	05h
_NVME_ASC_IDENTIFY	06h
_NVME_ASC_ABORT	08h
_NVME_ASC_SET_FEATURES	09h
_NVME_ASC_GET_FEATURES	0Ah
_NVME_ASC_ASYNC_EVENT_REQ	0Ch
_NVME_ASC_FIRMWARE_ACTIVATE	10h
_NVME_ASC_FIRMWARE_IMG_DWNLD	11h
_NVME_ASC_DEVICE_SELF_TEST	14h
_NVME_ASC_NAMESPACE_ATTACHMENT	15h
_NVME_ASC_KEEP_ALIVE	18h
_NVME_ASC_DIRECTIVE_SEND	19h
_NVME_ASC_DIRECTIVE_RECEIVE	1Ah
_NVME_ASC_VIRTUALIZATION_MANAGEMENT	1Ch
_NVME_ASC_NVME_MI_SEND	1Dh
_NVME_ASC_NVME_MI_RECEIVE	1Eh
_NVME_ASC_DOORBELL_BUFFER_CFG	7Ch
_NVME_ASC_FORMAT_NVM	80h
_NVME_ASC_SECURITY_SEND	81h
_NVME_ASC_SECURITY_RECEIVE	82h
_NVME_ASC_SANITIZE	84h
_NVME_ASC_GET_LBA_STATUS	85h

For _NVME_NVM_SUBMISSION_CMD transaction event:

Constant	**Value**
_NVME_NSC_FLUSH	00h
_NVME_NSC_WRITE	01h
_NVME_NSC_READ	02h
_NVME_NSC_WRITE_UNCORRECTABLE	04h
_NVME_NSC_COMPARE	05h
_NVME_NCS_WRITE_ZEROES	08h
_NVME_NSC_DATASET_MGMT	09h
_NVME_NSC_VERIFY	0Ch
_NVME_NSC_RESERVATION_REGISTER	0Dh
_NVME_NSC_RESERVATION_REPORT	0Eh

_NVME_NSC_RESERVATION_ACQUIRE	11h
_NVME_NSC_RESERVATION_RELEASE	15h
_NVME_NSC_ZONE_MGMT_SEND	79h
_NVME_NSC_ZONE_MGMT_RECEIVE	7Ah
_NVME_NSC_ZONE_APPEND	7Dh

in.nvme NSID: This field specifies the namespace ID that this command applies to.

in.nvme MPTR: Returns the address of a contiguous physical buffer of metadata or the address of an SGL segment containing exactly one SGL Descriptor which describes the metadata to transfer

in.nvme DPTR: This field specifies the data used in the command. The example of parsing DPTR for PRP and SGL data transfer is present in examp_nvme_submission_dptr.pevs file.

in.nvme CDW10: Returns command-specific Dword #10.

in.nvme CDW11: Returns command-specific Dword #11.

in.nvme CDW12: Returns command-specific Dword #12.

in.nvme CDW13: Returns command-specific Dword #13.

in.nvme CDW14: Returns command-specific Dword #14.

in.nvme CDW15: Returns command-specific Dword #15.

5.2.12.2 NVM transaction members specific to _NVME_ADMIN_COMPLETION_CMD and _NVME_NVM_COMPLETION_CMD event

in.nvmeCDW0: Returns command-specific Dword #0.

in.nvmeCDW1: Returns command-specific Dword #1.

in.nvmeSQID: Returns the Submission Queue to which the associated command was issued to.

in.nvmeSQHD: Returns the current Submission Queue Head pointer for the Submission Queue indicated in the SQ Identifier field.

in.nvmeSF: Returns the status for the command that is being completed.

in.nvmeSCT: Indicates the status code type of the completion queue entry.

in.nvmeSC: Indicates a status code identifying any error or status information for the command indicated.

in.nvmeP: Returns Phase Tag (P) (identifies whether a Completion Queue entry is new).

in.nvmeCID: Returns Command Id.

in.nvmeQID: Returns queue Id.

5.2.12.3 NVM transaction members specific to _NVME_PRP event

in.nvmeCID: Returns Command Id.

in.nvmeQID: Returns queue Id.

5.2.12.4 NVM transaction members specific to _NVME_SGL event

in.nvmeCID: Returns Command Id.

in.nvmeQID: Returns queue Id.

in.nvmeAddress: Returns address of the next SGL segment (64-bit value).

in.nvmeLength: Length of SGL segment.

in.nvmeSGLId: SGL identifier.

5.2.12.5 NVM transaction members specific to _NVME_TRANSFERED_DATA event

in.nvmeCID: Returns Command Id.

in.nvmeQID: Returns queue Id.

in.nvmeDataAddress: Returns SGL segment or PRP entry address.

in.nvmeDataLength: Length of SGL segment or PRP entry.

5.2.12.6 NVM transaction members specific to _NVME_INTERRUPT_REG event

in.nvmeInterruptType: Returns Interrupt Type.

in.nvmeInterruptVector: Returns Interrupt Vector Index.

in.nvmeInterruptMsg: Returns Interrupt Message.

5.2.12.7 Metric values

in.Metric_Throughput: Metric presenting transaction payload divided by response time, expressed in **kilobytes** per second, an integer value

in.Metric_PayloadBytes: Metric presenting number of data payload bytes this NVM Transaction transferred, an integer value.

in.Metric_NumOfLinkAndSplitTras: Metric presenting the total number of Link and Split Transactions that compose this NVM Transaction, an integer value.

5.2.13 NVM command-specific set of members

Valid for NVM commands only. Undefined for other events.

All the NVM command-specific values are present in the input context for NVM commands. Also the common PayloadLength and Payload values reflect the total combined payload for the NVM command. In addition to that, the following values exist:

in.nvmcDeviceId: Returns command device id.

in.nvmcCommandOpCode: Returns command opcode. The following possible values are defined by VSE and the corresponding constants can be used by scripts:

I / O commands		
Command	**Value**	**Command name**
_NVMC_OPCODE_FLUSH	0x00	Flush
_NVMC_OPCODE_WRITE	0x01	Write
_NVMC_OPCODE_READ	0x02	Read
_NVMC_OPCODE_WRITE_UNCORRECTABLE	0x04	Write Uncorrectable
_NVMC_OPCODE_COMPARE	0x05	Compare
_NVMC_OPCODE_WRITE_ZEROES	0x08	Write Zeros
_NVMC_OPCODE_DATASET_MGMT	0x09	Dataset Management
_NVMC_OPCODE_VERIFY	0x0C	Verify
_NVMC_OPCODE_RESERVATION_REGISTER	0x0D	Reservation Register
_NVMC_OPCODE_RESERVATION_REPORT	0x0E	Reservation Report
_NVMC_OPCODE_RESERVATION_ACQUIRE	0x11	Reservation Acquire
_NVMC_OPCODE_RESERVATION_RELEASE	0x15	Reservation Release
_NVMC_OPCODE_ZONE_MGMT_SEND	0x79	Zone Management Send
_NVMC_OPCODE_ZONE_MGMT_RECEIVE	0x7A	Zone Management Receive
_NVMC_OPCODE_ZONE_APPEND	0x7D	Zone Append
_NVMC_OPCODE_DATASET_MGMT_VENDOR_SPECIFIC_FIRST	0x80	Vendor Specific
_NVMC_OPCODE_DATASET_MGMT_VENDOR_SPECIFIC_LAST	0xFF	Vendor Specific
Admin commands		
_NVMC_OPCODE_DELETE_IO_SQ	0x00	Delete I/O Submission Queue
_NVMC_OPCODE_CREATE_IO_SQ	0x01	Create I/O Submission Queue
_NVMC_OPCODE_GET_LOG_PAGE	0x02	Get Log Page
_NVMC_OPCODE_DELETE_IO_CQ	0x04	Delete I/O Completion Queue
_NVMC_OPCODE_CREATE_IO_CQ	0x05	Create I/O Completion Queue
_NVMC_OPCODE_IDENTIFY	0x06	Identify
_NVMC_OPCODE_ABORT	0x08	Abort
_NVMC_OPCODE_SET_FEATURE	0x09	Set Features
_NVMC_OPCODE_GET_FEATURE	0x0A	Get Features
_NVMC_OPCODE_ASYNC_EVENT_REQUEST	0x0C	Asynchronous Event Request
_NVMC_OPCODE_NAMESPACE_MANAGEMENT	0x0D	Namespace Management
_NVMC_OPCODE_FIRWARE_ACTIVATE	0x10	Firmware Activate
_NVMC_OPCODE_FIRMWARE_IMG_DOWNLOAD	0x11	Firmware Image Download
_NVMC_OPCODE_DEVICE_SELF_TEST	0x14	Device Self-test
_NVMC_OPCODE_KEEP_ALIVE	0x18	Keep Alive
_NVMC_OPCODE_DIRECTIVE_SEND	0x19	Directive Send
_NVMC_OPCODE_DIRECTIVE_RECEIVE	0x1A	Directive Receive
_NVMC_OPCODE_VIRTUALIZATION_MANAGEMENT	0x1C	Virtualization Management
_NVMC_OPCODE_NVME_MI_SEND	0x1D	NVMe-MI Send

_NVMC_OPCODE_NVME_MI_RECEIVE	0x1E	NVMe_MI Receive
_NVMC_OPCODE_DOORBELL_BUFFER_CFG	0x7C	Doorbell Buffer Config
_NVMC_OPCODE_NAMESPACE_ATTACHMENT	0x15	Namespace Attachment
_NVMC_OPCODE_FORMAT_NVM	0x80	Format NVM
_NVMC_OPCODE_SECURITY_SEND	0x81	Security Send
_NVMC_OPCODE_SECURITY_RECEIVE	0x82	Security Receive
_NVMC_OPCODE_SANITIZE	0x84	Sanitize
_NVMC_OPCODE_GET_LBA_STATUS	0x85	Get LBA Status
_NVMC_OPCODE_OTHER_IO_COMMAND_SET_SPECIFIC	0x83	Other IO command set specific
_NVMC_OPCODE_VENDOR_SPECIFIC_FIRST	0xC0	Vendor Specific
_NVMC_OPCODE_VENDOR_SPECIFIC_LAST	0xFF	Vendor Specific

in.nvmcSubmissionQueueID: Returns command submission queue id.

in.nvmcCompletionQueueID: Returns command completion queue id.

in.nvmcCommandID: Returns command id.

in.nvmcIsSecurityCommand: If set to a non-zero value, indicates NVM command is security.

in.nvmcIsDeviceToHostCommand: If set to a non-zero value, indicates NVM command transfers data from device to host.

in.nvmcIsSuccessful: If set to a non-zero value, indicates NVM command is successful.

in.nvmcStatus: Returns command status numeric encoding.

in.nvmcStatusType: Returns command status type numeric encoding.

in.nvmcIsIncomplete: If set to a non-zero value, indicates NVM command is incomplete.

in.nvmcHasInput: NVM command contains submission queue entry.

in.nvmcHasOutput: NVM command contains completion queue entry.

in.nvmcIsAdminCommand: If set to a non-zero value, indicates NVM command is admin.

in.nvmcNamespaceId: Returns namespace Id.

in.nvmcErrorId: Returns NVM error with the smallest numeric encoding.

The following possible values are defined by VSE and the corresponding constants can be used by scripts:

Error type	Value	Error name	Error description
_NVMC_ERROR_NO_ERROR	0	No error	No errors
_NVMC_ERROR_INCOMPLETE_SUB_TRA	1	Incomplete Sub-Transaction	Incomplete Sub Transaction
_NVMC_ERROR_SUB_TRA_HAS_ERROR	2	Error in Sub-Transaction	Error in Sub Transaction
_NVMC_ERROR_INCOMPLETE_TRA	3	Incomplete Transaction	Incomplete command
_NVMC_ERROR_LOGICAL_ERROR	4	Logical Error	Logical error
_NVMC_ERROR_PAYLOAD_ERROR	5	Payload Error	Payload error

in.nvmcErrorIdAsString: Contains an NVM error name.

in.nvmcTraHasError: If set to a non-zero value, indicates NVM command has errors.

in.nvmcNumBytesTransferred: Returns amount of transferred data by command in bytes.

in.nvmcNumBytesRequested: Returns number bytes requested by command.

in.nvmcUtilizesPRP: If set to a non-zero value, indicates that command uses PRP instead of SGL.

in.nvmcHasInterrupt: NVM Command contains Interrupt.

in.nvmcInterruptType: Returns Interrupt Type.

in.nvmcInterruptVector: Returns Interrupt Vector Index.

in.nvmcInterruptMsg: Returns Interrupt Message.

The following table shows the list of NVM commands and their fields defined in the input context. The fields can be accessed by using "_". E.g. **in. CreateIOCQ_PC** contains the numeric encoding of the PC field of Create I/O Completion Queue command. Some commands have repeating blocks(Power State Descriptors in Identify command), these blocks can be accessed the following way: <Command>_<Block_name><index>_<fields>, i.e. **in.Identify_PSD3_MP**.

Note: If length of returned value is bigger than 1 dword, please, specify dword by using "_DW" and dword index, otherwise string in a hex format will be returned. For example **in. GetLogPage_POWER_CYCLES_DW0** contains the numeric encoding of the 1st dword of Power Cycles field of GetLogPage: SMART / Health Information Log command.

Submission Queue Entry Data

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
Abort	Abort	SQID	10	15:0	Submission Queue Identifier
		CID	10	31:16	Command Identifier
Create I/O Completion Queue	CreateIOCQ	PRP1	6	63:0	PRP Entry 1
		QID	10	15:0	Queue Identifier
		QSIZE	10	31:16	Queue Size
		PC	11	0	Physically Contiguous
		IEN	11	1	Interrupts Enabled
		RSVD	11	15:2	Reserved
		IV	11	31:16	Interrupt Vector
Create I/O Submission Queue	CreateIOSQ	PRP1	6	63:0	PRP Entry 1
		QID	10	15:0	Queue Identifier
		QSIZE	10	31:16	Queue Size
		PC	11	0	Physically Contiguous
		QPRIO	11	2:1	Queue Priority
		RSVD	11	15:3	Reserved
		CQID	11	31:16	Completion Queue Identifier
Delete I/O	DeleteIOCQ	QID	10	15:0	Queue Identifier

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
Completion Queue		RSVD	10	31:16	Reserved
Delete I/O Submission Queue	DeleteIOSQ	QID	10	15:0	Queue Identifier
		RSVD	10	31:16	Reserved
Device Self-test	DeviceSelftest	STC	10	4:0	Self-test Code
		RSVD	10	31:5	Reserved
Directive Receive	Directive	PRP1	6	63:0	PRP Entry 1
		PRP2	8	63:0	PRP Entry 2
		NUMD	10	31:0	Number of Dwords
		DOPER_IDEN_REC	11	7:0	Directive Operation (Identify)
		DOPER_STR_REC	11	7:0	Directive Operation (Streams)
		DTYPE	11	8:15	Directive Type
		DSPEC	11	31:16	Directive Specific
		NSR	12	15:0	NS Streams Requested
		RSVD	12	31:16	Reserved
Directive Send	Directive	PRP1	6	63:0	PRP Entry 1
		PRP2	8	63:0	PRP Entry 2
		NUMD	10	31:0	Number of Dwords
		DOPER_IDEN_SND	11	7:0	Directive Operation (Identify)
		DOPER_STR_SND	11	7:0	Directive Operation (Streams)
		DTYPE	11	8:15	Directive Type
		DSPEC	11	31:16	Directive Specific
		ENDIR	12	0:0	Enable Directive
		RSVD	12	7:1	Reserved
		DTYPE_EN_DIR	12	15:8	Directive Type
		RSVD2	12	31:16	Reserved
Doorbell Buffer Config	DoorbellBufferCfg	PRP1	6	63:0	PRP Entry 1
		PRP2	8	63:0	PRP Entry 2
Firmware Activate	FirmwareActivate	FS	10	2:0	Firmware Slot
		AA	10	4:3	Activate Action
		CA	10	4:3	Commit Action (NVMe 1.2)
		RSVD	10	30:05	Reserved
		BPID	10	31:31	Boot Partition ID
Firmware Image Download	FirmwareImageDownload	PRP1	6	63:0	PRP Entry 1
		PRP2	8	63:0	PRP Entry 2
		NUMD	10	31:0	Number of Dwords
		OFST	11	31:0	Offset
Get Features	GetFeatures	PRP1	6	63:0	PRP Entry 1
		PRP2	8	63:0	PRP Entry 2

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
		FID	10	7:0	Feature Identifier
		SEL	10	10:8	Select
		RSVD	10	31:11	Reserved
Get Log Page	GetLogPage	PRP1	6	63:0	PRP Entry 1
		PRP2	8	63:0	PRP Entry 2
		LID	10	7:0	Log Page Identifier
		LSP	10	11:8	Log Specific Field
		RSVD	10	14:11	Reserved
		RAE	10	15:15	Retain Asynchronous Event
		NUMDL	10	32:16	Number of Dwords Lower
		NUMDU	11	15:0	Number of Dwords Upper
		RSVD1	11	31:16	Reserved (NVMe 1.3)
		LSI	11	31:16	Log Specific Identifier (NVMe 1.4)
		LPOL	12	32:0	Log Page Offset Lower
		LPOU	13	32:0	Log Page Offset Upper
		UUID_IDX	14	6:0	UUID Index (NVMe 1.4)
		RSVD2	14	23:7	Reserved (NVMe 1.4)
		CSI	14	31:24	Command Set Identifier (NVMe 1.4)
Identify	Identify	PRP1	6	63:0	PRP Entry 1
		PRP2	8	63:0	PRP Entry 2
		CNS	10	1:0	Controller or Namespace Structure
		RSVD	10	31:2	Reserved
		CNS	10	7:0	Controller or Namespace Structure (NVMe 1.2)
		RSVD	10	15:8	Reserved (NVMe 1.2)
		CNTID	10	31:16	Controller Identifier (NVMe 1.2)
		NVMSETID	11	15:0	NVM Set Identifier
		RSVD1	11	23:16	Reserved
		CSI	11	31:24	Command Set Identifier (NVMe 1.4)
		UUID_IDX	12	6:0	UUID Index
		RSVD2	12	31:7	Reserved
Namespace Attachment	NamespaceAttachment	PRP1	6	63:0	PRP Entry 1
		PRP2	8	63:0	PRP Entry 2
		SEL	10	3:0	Select
		RSVD	10	31:4	Reserved
Namespace Management	NamespaceManagement	PRP1	6	63:0	PRP Entry 1
		PRP2	8	63:0	PRP Entry 2
		SEL	10	3:0	Select

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
		RSVD	10	31:4	Reserved
		RSVD1	11	23:0	Reserved
		CSI	11	31:24	Command Set Identifier (NVMe 1.4)
NVMe-MI Send	NVMeMISend	PRP1	6	63:0	PRP Entry 1
		PRP2	8	63:0	PRP Entry 2
		NVME_MI_SEND_NMSP0	10	7:0	NVMe-MI Specific 0
		RSVD	10	31:8	Reserved
		NVME_MI_SEND_NUMD	11	31:0	Number of Dwords
NVMe-MI Receive	NVMeMIReceive	PRP1	6	63:0	PRP Entry 1
		PRP2	8	63:0	PRP Entry 2
		NVME_MI_RECEIVE_NMSP0	10	7:0	NVMe-MI Specific 0
		RSVD	10	31:8	Reserved
Set Features	SetFeatures	PRP1	6	63:0	PRP Entry 1
		PRP2	8	63:0	PRP Entry 2
		FID	10	7:0	Feature Identifier
		RSVD	10	30:8	Reserved
		SV	10	31	Save
Format NVM	FormatNVM	LBAF	10	3:0	LBA Format
		MS, MSET	10	4	Metadata Settings
		PI	10	7:5	Protection Information
		PIL	10	8	Protection Information Location
		SES	10	11:9	Secure Erase Settings
		RSVD	10	31:12	Reserved
Security Receive	SecurityReceive	PRP1	6	63:0	PRP Entry 1
		PRP2	8	63:0	PRP Entry 2
		RSVD	10	7:0	Reserved
		NSSF	10	7:0	NVMe Security Specific Field (NVMe 1.2)
		SPSP	10	23:8	SP Specific
		SECP	10	31:24	Security Protocol
		AL	11	31:0	Allocation Length
Security Send	SecuritySend	PRP1	6	63:0	PRP Entry 1
		PRP2	8	63:0	PRP Entry 2
		RSVD	10	7:0	Reserved
		NSSF	10	7:0	NVMe Security Specific Field (NVMe 1.2)
		SPSP	10	23:8	SP Specific
		SECP	10	31:24	Security Protocol

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
		TL	11	31:0	Transfer Length
Virtualization Management	VirtualizationManagement	VIRTUALIZATION_MANAGEMENT_ACT	10	3:0	Action
		RSVD	10	7:4	Reserved
		VIRTUALIZATION_MANAGEMENT_RT	10	10:8	Resource Type
		RSVD1	10	15:11	Reserved
		VIRTUALIZATION_MANAGEMENT_CNTLID	10	31:16	Controller ID
		VIRTUALIZATION_MANAGEMENT_NR	11	15:0	Number of Controller Resources
		RSVD2	11	31:16	Reserved
Sanitize	Sanitize	SANITIZE_SANACT	10	2:0	Sanitize Action
		AUSE	10	3:3	Allow Unrestricted Sanitize Exit
		OWPASS	10	7:4	Overwrite Pass Count
		OIPBP	10	8:8	Overwrite Invert Pattern Between Passes
		NDAS	10	9:9	No Deallocate After Sanitize
		RSVD	10	31:10	Reserved
		OVRPAT	11	31:0	Overwrite Pattern
Get LBA Status	Get_LBA_Status	PRP1	6	63:0	PRP Entry 1
		PRP2	8	63:0	PRP Entry 2
		SLBA	10	63:0	Starting LBA
		MNDW	12	31:0	Maximum Number of Dwords
		RL	13	15:0	Range Length
		RSVD	13	23:16	Reserved
		ATYPE	13	31:24	Action Type
Read	Read	MPTR	4	63:0	Metadata Pointer
		PRP1	6	63:0	PRP Entry 1
		PRP2	8	63:0	PRP Entry 2
		MSGLP	4	63:0	Metadata SGL Segment Pointer
		SGL1	6	63:0	SGL Entry 1
		SLBA	10	63:0	Starting LBA
		NLB	12	15:0	Number of Logical Blocks
		RSVD	12	25:16	Reserved
		PRINFO	12	29:26	Protection Information Field
		FUA	12	30	Force Unit Access
		LR	12	31	Limited Retry

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
		DSM	13	7:0	Dataset Management
		ACCF	13	3:0	Access Frequency
		ACCL	13	5:4	Access Latency
		SEQR	13	6	Sequential Request
		INCOM	13	7	Incompressible
		RSVD1	13	10:8	Reserved
		EILBRT	14	31:0	Expected Initial Logical Block Reference Tag
		ELBAT	15	15:0	Expected Logical Block Application Tag
		ELBATM	15	31:16	Expected Logical Block Application Tag Mask
Reservation Acquire	ReservationAcquire	PRP1	6	63:0	PRP Entry 1
		PRP2	8	63:0	PRP Entry 2
		SGL1	6	63:0	SGL Entry 1
		RACQA	10	2:0	Reservation Acquire Action
		IEKEY	10	3	Ignore Existing Key
		RSVD	10	7:4	Reserved
		RTYPE	10	15:8	Reservation Type
		RSVD1	10	31:16	Reserved
Reservation Register	ReservationRegister	PRP1	6	63:0	PRP Entry 1
		PRP2	8	63:0	PRP Entry 2
		SGL1	6	63:0	SGL Entry 1
		RREGA	10	2:0	Reservation Register Action
		IEKEY	10	3	Ignore Existing Key
		RSVD	10	29:4	Reserved
		CPTPL	10	31:30	Change Persist Through Power Loss State
Reservation Release	ReservationRelease	PRP1	6	63:0	PRP Entry 1
		PRP2	8	63:0	PRP Entry 2

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
		SGL1	6	63:0	SGL Entry 1
		RRELA	10	2:0	Reservation Release Action
		IEKEY	10	3	Ignore Existing Key
		RSVD	10	7:4	Reserved
		RTYPE	10	15:8	Reservation Type
		RSVD1	10	31:16	Reserved
Reservation Report	ReservationReport	PRP1	6	63:0	PRP Entry 1
		PRP2	8	63:0	PRP Entry 2
		SGL1	6	63:0	SGL Entry 1
		NUMD	10	31:16	Number of Dwords
		EDS	11	0:0	Extended Data Structure
		RSVD	11	31:1	Reserved
Write	Write	MPTR	4	63:0	Metadata Pointer
		PRP1	6	63:0	PRP Entry 1
		PRP2	8	63:0	PRP Entry 2
		MSGLP	4	63:0	Metadata SGL Segment Pointer
		SGL1	6	127:0	SGL Entry 1
		SLBA	10	63:0	Starting LBA
		NLB	12	15:0	Number of Logical Blocks
		RSVD	12	19:16	Reserved
		DTYPE	12	23:20	Directive Type
		RSVD1	12	25:24	Reserved
		PRINFO	12	29:26	Protection Information Field
		FUA	12	30	Force Unit Access
		LR	12	31	Limited Retry
		DSM	13	7:0	Dataset Management
		ACCF	13	3:0	Access Frequency
		ACCL	13	5:4	Access Latency
		SEQR	13	6	Sequential Request
		INCOM	13	7	Incompressible
		RSVD1	13	15:8	Reserved
		DSPEC	13	31:16	Directive Specific
		ILBRT	14	31:0	Initial Logical Block Reference Tag
		LBAT	15	15:0	Logical Block Application Tag
		LBATM	15	31:16	Logical Block Application Tag Mask
Write Uncorrectable	WriteUncorrectable	SLBA	10	63:0	Starting LBA
		NLB	12	15:0	Number of Logical Blocks
		RSVD	12	31:16	Reserved

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
Compare	Compare	MPTR	4	63:0	Metadata Pointer
		PRP1	6	63:0	PRP Entry 1
		PRP2	8	63:0	PRP Entry 2
		MSGLP	4	63:0	Metadata SGL Segment Pointer
		SGL1	6	127:0	SGL Entry 1
		SLBA	10	63:0	Starting LBA
		NLB	12	15:0	Number of Logical Blocks
		RSVD	12	25:16	Reserved
		PRINFO	12	29:26	Protection Information Field
		FUA	12	30	Force Unit Access
		LR	12	31	Limited Retry
		EILBRT	14	31:00	Expected Initial Logical Block Reference Tag
		ELBAT	15	15:00	Expected Logical Block Application Tag
		ELBATM	15	31:16	Expected Logical Block Application Tag Mask
Dataset Management	DatasetManagement	MPTR	4	63:0	Metadata Pointer
		PRP1	6	63:0	PRP Entry 1
		PRP2	8	63:0	PRP Entry 2
		MSGLP	4	63:0	Metadata SGL Segment Pointer
		SGL1	6	127:0	SGL Entry 1
		NR	10	7:0	Number of Ranges
		RSVD	10	31:8	Reserved
		IDR	11	0	Attribute - Integral Dataset for Read
		IDW	11	1	Attribute - Integral Dataset for Write
		AD	11	2	Attribute - Deallocate
		RSVD1	11	31:3	Reserved
Write Zeroes	WriteZeros	SLBA	10	63:0	Starting LBA
		NLB	12	15:0	Number of Logical Blocks
		RSVD	12	24:16	Reserved
		DEAC	12	25:25	Deallocate
		PRINFO	12	29:26	Protection Information Field
		FUA	12	30	Force Unit Access
		LR	12	31	Limited Retry

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
		ILBRT	14	31:0	Initial Logical Block Reference Tag
		LBAT	15	15:0	Logical Block Application Tag
		LBATM	15	31:16	Logical Block Application Tag Mask
Verify	Verify	SLBA	10	63:0	Starting LBA
		NLB	12	15:0	Number of Logical Blocks
		RSVD	12	25:16	Reserved
		PRINFO	12	29:26	Protection Information Field
		FUA	12	30	Force Unit Access
		LR	12	31	Limited Retry
		EILBRT	14	31:0	Expected Initial Logical Block Reference Tag
		ELBAT	15	15:0	Expected Logical Block Application
		ELBATM	15	31:16	Expected Logical Block Application Tag Mask
Zone Management Send	ZoneManagementSend	PRP1	6	63:0	PRP Entry 1
		PRP2	8	63:0	PRP Entry 2
		SLBA	10	63:0	Starting LBA (NVMe 1.4)
		ZSA	13	7:0	Zone Send Action (NVMe 1.4)
		SEL_ALL	13	8	Select All
		RSVD	13	31:9	Reseved
Zone Management Receive	ZoneManagementReceive	PRP1	6	63:0	PRP Entry 1
		PRP2	8	63:0	PRP Entry 2
		SLBA	10	63:0	Starting LBA (NVMe 1.4)
		NUMD	12	31:0	Number of Dwords (NVMe 1.4)
		ZRA	13	7:0	Zone Receive Action (NVMe 1.4)
		ZRASFD	13	15:8	Zone Receive Action Specific Field (NVMe 1.4)
		ZRASFT	13	16	Zone Receive Action Specific Features (NVMe 1.4)
		RSVD	13	31:17	Reserved
Zone Append	ZoneAppend	MPTR	4	63:0	Metadata Pointer
		PRP1	6	63:0	PRP Entry 1
		PRP2	8	63:0	PRP Entry 2
		MSGLP	4	63:0	Metadata SGL Segment Pointer
		SGL1	6	127:0	SGL Entry 1
		ZSLBA	10	63:0	Zone Start LBA
		NLB	12	15:0	Number of Logical Blocks
		RSVD	12	24:16	Reserved

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
		PIREMAP	12	25	Protection Information Remap
		PRINFO	12	29:26	Protection Information Field
		FUA	12	30	Force Unit Access
		LR	12	31	Limited Retry
		ILBRT	14	31:0	Initial Logical Block Reference Tag
		LBAT	15	15:0	Logical Block Application Tag
		LBATM	15	31:16	Logical Block Application Tag Mask
Get / Set Features: Arbitration	GetFeatures / SetFeatures	AB	11	2:0	Arbitration Burst
		RSVD	11	7:3	Reserved
		LPW	11	15:8	Low Priority Weight
		MPW	11	23:16	Medium Priority Weight
		HPW	11	31:24	High Priority Weight
Get / Set Features: Power Management	GetFeatures / SetFeatures	PS	11	4:0	Power State
		WH	11	7:5	Workload Hint
		RSVD	11	31:8	Reserved
Get / Set Features: LBA Range Type	GetFeatures_LBARangeType<index> / SetFeatures_LBARangeType<index>	NUM	11	5:0	Number of LBA Ranges
		RSVD	11	31:6	Reserved
Get / Set Features: Temperature Threshold	GetFeatures / SetFeatures	TMPTH	11	15:0	Temperature Threshold
		TMPSEL	11	19:16	Threshold Temperature Select
		THSEL	11	21:20	Threshold Type Select
		RSVD	11	31:22	Reserved
Get / Set Features: Error Recovery	GetFeatures / SetFeatures	TLER	11	15:0	Time Limited Error Recovery
		DULBE	11	16:16	Deallocated or Unwritten Logical Block Error Enable
		RSVD	11	31:17	Reserved
Get / Set Features: Volatile Write Cache	GetFeatures / SetFeatures	WCE	11	0	Volatile Write Cache Enable
		RSVD	11	31:1	Reserved
Get / Set Features: Number of Queues	GetFeatures / SetFeatures	NSQR	11	15:0	Number of I/O Submission Queues Requested
		NCQR	11	31:16	Number of I/O Completion Queues Requested
Get / Set Features: Interrupt Coalescing	GetFeatures / SetFeatures	THR	11	7:0	Aggregation Threshold
		TIME	11	15:8	Aggregation Time
		RSVD	11	31:16	Reserved
Get / Set Features:	GetFeatures / SetFeatures	IV	11	15:0	Interrupt Vector
		CD	11	16	Coalescing Disable

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
Interrupt Vector Configuration		RSVD	11	31:17	Reserved
Keep Alive Timer	GetFeatures / SetFeatures	KATO	11	31:0	Keep Alive Timeout
Host Controlled Thermal Management	GetFeatures / SetFeatures	TMT2	11	15:0	Thermal Management Temperature 2
		TMT1	11	31:16	Thermal Management Temperature 1
Non-Operational Power State Config	GetFeatures / SetFeatures	NOPPME	11	0:0	Non-Operational Power State Permissive Mode Enable
		RSVD	11	31:1	Reserved
Get Features: Recovery Level Config	GetFeatures	NVMSETID	11	15:0	NVM Set Identifier
		RSVD	11	31:16	Reserved
Set Features: Recovery Level Config	SetFeatures	NVMSETID	11	15:0	NVM Set Identifier
		RSVD	11	31:16	Reserved
		RRL	12	3:0	Read Recovery Level
		RSVD2	12	31:4	Reserved
Get Features: Predictable Latency Mode Config	GetFeatures	NVMSETID	11	15:0	NVM Set Identifier
		RSVD	11	31:16	Reserved
Set Features: Predictable Latency Mode Config	SetFeatures	NVMSETID	11	15:0	NVM Set Identifier
		RSVD	11	31:16	Reserved
		PLE	12	0	Predictable Latency Enable
		RSVD2	12	31:1	Reserved
Get Features: Predictable Latency Mode Window	GetFeatures	NVMSETID	11	15:0	NVM Set Identifier
		RSVD	11	31:16	Reserved
Set Features: Predictable Latency Mode Window	SetFeatures	NVMSETID	11	15:0	NVM Set Identifier
		RSVD	11	31:16	Reserved
		WS	12	2:0	Window Select
		RSVD2	12	31:3	Reserved
Set Features: LBA Status Information Attributes	SetFeatures	LSIRI	11	15:0	LBA Status Information Report Interval
		LSIPI	11	31:16	LBA Status Information Poll Interval
Set Features: Sanitize Config	SetFeatures	NODRM	11	0	No-Deallocate Response Mode
		RSVD	11	31:1	Reserved
Get Features: Endurance Group Event Config	GetFeatures	ENDGID	11	15:0	Endurance Group Identifier
Set Features: Endurance	SetFeatures	ENDGID	11	15:0	Endurance Group Identifier
		EGCW_BIT0	11	16	Endurance Group Critical

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
Group Event Config					Warnings Bit0
		RSVD	11	17	Reserved
		EGCW_BIT2	11	18	Endurance Group Critical Warnings Bit2
		EGCW_BIT3	11	19	Endurance Group Critical Warnings Bit3
		RSVD2	11	23:20	Reserved
		RSVD3	11	31:24	Reserved
Get / Set Features: Write Atomicity	GetFeatures / SetFeatures	DN	11	0	Disable Normal
		RSVD	11	31:1	Reserved
Get / Set Features: Asynchronous Event Configuration	GetFeatures / SetFeatures	SMART	11	7:0	SMART / Health Critical Warnings
		NAN	11	8	Namespace Attribute Notices
		FAN	11	9	Firmware Activation Notices
		TLN	11	10	Telemetry Log Notices
		ANACN	11	11	Asymmetric Namespace Access Change Notices
		PLEALCN	11	12	Predictable Latency Event Aggregate Log Change Notices
		LBASIN	11	13	LBA Status Information Notices
		EGEALCN	11	14	Endurance Group Event Aggregate Log Change Notices
		RSVD	11	26:15	Reserved
		ZDCHN	11	27	Zone Descriptor Changed Notices
		RSVD1	11	31:28	Reserved
Get / Set Features: Autonomous Power State Transition	GetFeatures / SetFeatures	APSTE	11	0	Autonomous Power State Transition Enable
		RSVD	11	31:1	Reserved
Get / Set Features: Host Memory Buffer	GetFeatures / SetFeatures	EHM	11	0	Enable Host Memory
		MR	11	1	Memory Return
		RSVD	11	31:02	Reserved
		HSIZE	12	31:00	Host Memory Buffer Size
		HMDLLA	13	31:00	Host Memory Descriptor List Lower Address
		HMDLUA	14	31:00	Host Memory Descriptor List Upper Address
		HMDLEC	15	31:00	Host Memory Descriptor List Entry Count
Host Identifier	GetFeatures / SetFeatures	EXHID	11	0:0	Enable Extended Host Identifier
		RSVD	11	31:1	Reserved

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
Get / Set Features: Software Progress Marker	GetFeatures / SetFeatures	PBSLC	11	7:0	Pre-boot Software Load Count
		RSVD	11	31:8	Reserved
Get / Set Features: Reservation Notification Configuration	GetFeatures / SetFeatures	RSVD	11	0	Reserved
		REGPRE	11	1	Mask Registration Preempted Notification
		RESREL	11	2	Mask Reservation Released Notification
		RESPRE	11	3	Mask Reservation Preempted Notification
		RSVD1	11	31:4	Reserved
Get / Set Features: Reservation Persistence	GetFeatures / SetFeatures	PTPL	11	0	Persist Through Power Loss
		RSVD	11	31:1	Reserved
Set Features: Namespace Write Protection Config	SetFeatures	WPS	11	2:0	Write Protection State
		RSVD	11	31:3	Reserved
Set Features: I/O Command Set Profile	SetFeatures	IOCSCI	11	8:0	I/O Command Set Combination Index
		RSVD	11	31:9	Reserved

Completion Queue Entry Data

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
Asynchronous Event Request	AsyncEventRequest	AE_TYPE	0	2:0	Asynchronous Event Type
		RSVD	0	7:3	Reserved
		AE_INFO	0	15:8	Asynchronous Event Information
		ASSOCIATED_LOG_PAGE	0	23:16	Associated Log Page
		RSVD1	0	31:24	Reserved
		NSID	1	31:0	Namespace Identifier
Directive Send	Directive	NSA	0	15:0	Namespace Streams Allocated
		RSVD	0	31:16	Reserved
Get / Set Features: Number of Queues	GetFeatures / SetFeatures	NSQA	0	15:0	Number of I/O Submission Queues Allocated
		NCQA	0	31:16	Number of I/O Completion Queues Allocated

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
Get Features: Keep Alive Timer	GetFeatures	KATO	0	31:0	Keep Alive Timeout
Get Features: Host Controlled Thermal Management	GetFeatures	TMT2	0	15:0	Thermal Management Temperature 2
		TMT1	0	31:16	Thermal Management Temperature 1
Get Features: Non Operational Power State Config	GetFeatures	NOPPME	0	0:0	Non-Operational Power State Permissive Mode Enable
		RSVD	0	31:1	Reserved
Get Features: Read Recovery Level Config	GetFeatures	RRLC_RRL	0	3:0	Read Recovery Level
		RSVD	0	31:4	Reserved
Get Features: Predictable Latency Mode Config	GetFeatures	PLE	0	0	Predictable Latency Enable
		RSVD	0	31:1	Reserved
Get Features: Predictable Latency Mode Window	GetFeatures	WS	0	2:0	Window Select
		RSVD		31:3	Reserved
Get / Set Features: LBA Status Information Attributes	GetFeatures / SetFeatures	LSIRI	0	15:0	LBA Status Information Report Interval
		LSIPI	0	31:16	LBA Status Information Poll Interval
Get Features: Sanitize Config	GetFeatures	NODRM	0	0	No-Deallocate Response Mode
		RSVD	0	31:1	Reserved
Get Features: Endurance Group Event Config	GetFeatures	ENDGID	0	15:0	Endurance Group Identifier
		EGCW_BIT0	0	16	Endurance Group Critical Warnings Bit0
		RSVD	0	17	Reserved
		EGCW_BIT2	0	18	Endurance Group Critical Warnings Bit2
		EGCW_BIT3	0	19	Endurance Group Critical Warnings Bit3
		RSVD2	0	23:20	Reserved
		RSVD3	0	31:24	Reserved
Get Features	GetFeatures	WPS	0	2:0	Write Protection State
		RSVD	0	31:3	Reserved
Get Features: Interrupt	GetFeatures	THR	0	7:0	Aggregation Threshold

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
Coalescing		TIME	0	15:8	Aggregation Time
		RSVD	0	31:16	Reserved
Get Features: Host Memory Buffer	GetFeatures	EHM	0	0	Enable Host Memory
		MR	0	1	Memory Return
		RSVD	0	31:02	Reserved
Get Features: I/O Command Set Profile	GetFeatures	IOCSCI	0	8:0	I/O Command Set Combination Index
		RSVD	0	31:9	Reserved
Namespace Management	NamespaceManagement	NSID	0	31:00	Namespace Identifier
Virtualization Management	VirtualizationManagement	NRM	0	15:00	Number of Controller Resources Modified
Zone Management Send	ZoneManagementSend	ZCC	0	0	Zone Capacity Changed
		RSVD	0	31:1	Reserved
Zone Append	ZoneAppend	LBA	0	63:0	The lowest LBA for the command

Payload Data

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
Get / Set Features: LBA Range Type	GetFeatures / SetFeatures	TYPE	0	7:0	Attributes
		ATTRIBUTES_BIT0	0	8	Attributes bit 0
		ATTRIBUTES_BIT1	0	9	Attributes bit 1
		RSVD	0	15:10	Reserved
		RSVD1	0	127:16	Reserved
		SLBA	4	63:0	Starting LBA
		NLB	6	63:0	Number of logical blocks
		GUID	8	127:0	Unique Identifier
		RSVD2	12	127:0	Reserved
Get / Set Features: Autonomous Power State	GetFeatures / SetFeatures	RSVD	0	2:0	Reserved
		ITPS	0	7:3	Idle Transition Power State
		ITPT	0	31:8	Idle Time Prior to Transition

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
Transition		RSVD1	1	31:0	Reserved
Get Features: Host Memory Buffer	GetFeatures	HSIZE	0	31:0	Host Memory Buffer Size
		HMDLLA	1	31:0	Host Memory Descriptor List Lower Address
		HMDLUA	2	31:0	Host Memory Descriptor List Upper Address
		HMDLEC	3	31:0	Host Memory Descriptor List Entry Count
Set Features: Timestamp	SetFeatures	TIMESTAMP	0	47:0	Timestamp
		RSVD	0	63:48	Reserved
Get Features: Timestamp	GetFeatures	TIMESTAMP	0	47:0	Timestamp
		SYNCH	0	48:48	Synch
		ORIGIN	0	52:49	Timestamp Origin
		RSVD	0	56:53	Reserved
		RSVD1	0	63:57	Reserved
Get / Set Features: Host Identifier	GetFeatures / SetFeatures	HOSTID	0	63:0	Host Identifier
Get Features: Predictable Latency Mode Config	GetFeatures	EE_BIT0	0	0	Enable Event Bit0
		EE_BIT1	0	1	Enable Event Bit1
		EE_BIT2	0	2	Enable Event Bit2
		RSVD	0	13:3	Reserved
		EE_BIT14	0	14	Enable Event Bit14
		EE_BIT15	0	15	Enable Event Bit15
		RSVD2	0	255:16	Reserved
		DTWIN_RD_THR	8	63:0	DTWIN Reads Threshold
		DTWIN_WR_THR	10	63:0	DTWIN Writes Threshold
		DTWIN_TIME_THR	12	63:0	DTWIN Time Threshold
		RSVD3	14	3647:0	Reserved
Get / Set Features: Host Behavior Support	GetFeatures / SetFeatures	ACRE	0	0	Advanced Command Retry Enable
		RSVD	0	511:1	Reserved
Directive Receive (DTYPE: Identify DOPER: Return Parameters)	Directive	IDENTIFY_SUP	0	0:0	Directives Supported
		STREAMS_SUP	0	1:1	Directives Supported
		RSVD	0	255:2	Reserved
		IDENTIFY_EN	8	0:0	Directives Enabled
		STREAMS_EN	8	1:1	Directives Enabled
		RSVD1	8	255:2	Reserved
		RSVD2	16	32256:0	Reserved
Directive Receive	Directive	MSL	0	15:0	Max Streams Limit
		NSSA	0	31:16	NVM Subsystem Streams

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
(DTYPE: Streams DOPER: Return Parameters)					Available
		NSSO	1	15:0	NVM Subsystem Streams Open
		RSVD	1	95:16	Reserved
		SWS	4	31:0	Stream Write Size
		SGS	5	15:0	Stream Granularity Size
		NSA	5	31:16	Namespace Streams Allocated
		NSO	6	15:0	Namespace Streams Open
		RSVD1	6	63:16	Reserved
Directive Receive (DTYPE: Streams DOPER: Get Status	Directive_Streams<index>	CNT	0	15:0	Open Stream Count
		STREAM_ID	0	31:16	Stream Identifier
GetLogPage: Error Information	GetLogPage	ERROR_COUNT	0	63:0	Error Count
		SUB_Q_ID	2	15:0	Submission Queue ID
		COMMAND_ID	2	31:16	Command ID
		STATUS_FIELD	3	15:0	Status Field
		BYTE_WITH_ERROR	3	23:16	Parameter Error Location: Byte
		BIT_WITH_ERROR	3	26:24	Parameter Error Location: Bit
		RSVD	3	31:27	Reserved
		LBA	4	63:0	LBA
		NAMESPACE	6	31:0	Namespace
		VENDOR_SPECIFIC_INFO	7	7:0	Vendor Specific Information Available
		TRAN_TYPE	7	8:15	Transport Type (TRTYPE) (NVMe 1.4)
		RSVD1	7	16:31	Reserved (NVMe 1.4)
		COMMAND_SPECIFIC_INFO	8	63:0	Command Specific Information
		TRAN_TYPE_SPEC_INFO	10	15:0	Transport Type Specific Information (NVMe 1.4)
		RSVD2	10	511:16	Reserved (NVMe 1.4)
		RSVD1	7	271:8	Reserved (NVMe 1.3)
GetLogPage: Commands Supported and Effects	GetLogPage_CommandEffects<index>	CSUPP	0	0	Command Supported
		LBCC	0	1	Logical Block Content Change
		NCC	0	2	Namespace Capability Change
		NIC	0	3	Namespace Inventory Change

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
		CCC	0	4	Controller Capability Change
		CSE	0	18:16	Command Submission and Execution
		UUID_SS	0	19	UUID Selection Supported (NVMe 1.4)
GetLogPage: SMART / Health Information Log	GetLogPage	CRITICAL_WARNING_BIT0	0	0	Critical Warning Bit 0
		CRITICAL_WARNING_BIT1	0	1	Critical Warning Bit 1
		CRITICAL_WARNING_BIT2	0	2	Critical Warning Bit 2
		CRITICAL_WARNING_BIT3	0	3	Critical Warning Bit 3
		CRITICAL_WARNING_BIT4	0	4	Critical Warning Bit 4
		CRITICAL_WARNING_BIT5	0	5	Critical Warning Bit 5 (NVMe 1.4)
		RSVD	0	7:5	Reserved (NVMe 1.3)
		RSVD	0	7:6	Reserved (NVMe 1.4)
		TEMPERATURE	0	23:8	Temperature
		AVAILABLE_SPARE	0	31:24	Available Spare
		AVAILABLE_SPARE_THRESHOLD	1	7:0	Available Spare Threshold
		PERCENTAGE_USED	1	15:8	Percentage Used
		RSVD1	1	223:16	Reserved (NVMe 1.3)
		ENDURANCE_GROUP_CRITICAL_WARNING_BIT0	1	16	Endurance Group Critical Warning Summary Bit 0 (NVMe 1.4)
		RSVD3	1	17	Reserved Bit (NVMe 1.4)
		ENDURANCE_GROUP_CRITICAL_WARNING_BIT2	1	18	Endurance Group Critical Warning Summary Bit 2 (NVMe 1.4)
		ENDURANCE_GROUP_CRITICAL_WARNING_BIT3	1	19	Endurance Group Critical Warning Summary Bit 3 (NVMe 1.4)
		RSVD1	1	223:20	Reserved (NVMe 1.4)
		DATA_UNITS_READ	8	127:0	Data Units Read

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
		DATA_UNITS_WRITTEN	12	127:0	Data Units Written
		HOST_READ_COMMANDS	16	127:0	Host Read Commands
		HOST_WRITE_COMMANDS	20	127:0	Host Write Commands
		CONTROLLER_BUSY_TIME	24	127:0	Controller Busy Time
		POWER_CYCLES	28	127:0	Power Cycles
		POWER_ON_HOURS	32	127:0	Power On Hours
		UNSAFE_SHUTDOWNS	36	127:0	Unsafe Shutdowns
		MEDIA_ERRORS	40	127:0	Media Errors
		NUMBER_OF_ERROR_INFORMATION_LOG_ENTRIES	44	127:0	Number of Error Information Log Entries
		WCTT	48	31:0	Warning Composite Temperature Time
		CCTT	49	31:0	Critical Composite Temperature Time
		TS1	50	15:0	Temperature Sensor 1
		TS2	50	31:16	Temperature Sensor 2
		TS3	51	15:0	Temperature Sensor 3
		TS4	51	31:16	Temperature Sensor 4
		TS5	52	15:0	Temperature Sensor 5
		TS6	52	31:16	Temperature Sensor 6
		TS7	53	15:0	Temperature Sensor 7
		TS8	53	31:16	Temperature Sensor 8
		TMT1TC	54	31:0	Thermal Management Temperature 1 Transition Count
		TMT2TC	55	31:0	Thermal Management Temperature 2 Transition Count
		TTFTMT1	56	31:0	Total Time For Thermal Management Temperature 1
		TTFTMT2	57	31:0	Total Time For Thermal Management Temperature 2
		RSVD2	58	2240:0	Reserved
GetLogPage: Firmware Slot Information	GetLogPage	AFI_BITS0_2	0	2:0	Active Firmware Info Bits 2:0
		RSVD	0	3	Reserved
		AFI_BITS4_6	0	6:4	Active Firmware Info Bits 6:4
		RSVD1	0	7	Reserved

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
		RSVD2	0	63:8	Reserved
		FRS1	2	63:0	Firmware Revision for Slot 1
		FRS2	4	63:0	Firmware Revision for Slot 2
		FRS3	6	63:0	Firmware Revision for Slot 3
		FRS4	8	63:0	Firmware Revision for Slot 4
		FRS5	10	63:0	Firmware Revision for Slot 5
		FRS6	12	63:0	Firmware Revision for Slot 6
		FRS7	14	63:0	Firmware Revision for Slot 7
		RSVD3	16	3583:0	Reserved
GetLogPage: Device Self-test	GetLogPage_ <index>	CDST_COMPL	0	31:16	Current Device Self-Test Completion
		CDST_OPER	0	15:0	Current Device Self-Test Operation
GetLogPage: Device Self-test	GetLogPage_DeviceSelfTest<index>	CDST_OPER_BITS0_3	0	3:0	Current Device Self-Test Operation Bits 3:0
		RSVD	0	7:4	Reserved
		CDST_COMPL_BITS0_6	0	14:8	Current Device Self-Test Completion Bits 6:0
		RSVD1	0	15:15	Reserved
		RSVD2	0	31:16	Reserved
		Repeating structure:			
		DST_STATUS	0	15:0	Device Self-test Status
		DST_STATUS_BITS0_3	0	3:0	Device Self-test Status Bits 3:0
		DST_STATUS_BITS4_7	0	7:4	Device Self-test Status Bits 7:4
		SEGMENT_NUMBER	0	15:8	Segment Number
		VALID_DIAG_INFO	0	15:0	Valid Diagnostic Information
		NSID_VALID	0	16:16	NSID_Valid
		FLBA_VALID	0	17:17	FLBA_Valid
		SCT_VALID	0	18:18	SCT_Valid
		SC_VALID	0	19:19	SC_Valid
		RSVD3	0	23:20	Reserved
		RSVD4	0	31:25	Reserved
		POH	1	63:0	Power On Hours
		NSID	3	31:0	Namespace Identifier
		FAILING_LBA	4	63:0	Failing LBA
		STATUS_CODE_TYPE	6	15:0	Status Code Type
		STATUS_CODE_TYPE_BITS0_2	6	2:0	Status Code Type Bits 2:0
		RSVD5	6	7:3	Reserved
		STATUS_CODE	6	15:8	Status Code
		VENDOR_SPECIFIC	6	31:16	Vendor Specific
GetLogPage:	GetLogPage	THI_LOG_IDE	0	7:0	Log Identifier

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
Telemetry Host Initiated		NTIFIER			
		RSVD	0	39:8	Reserved
		IEEE	1	31:8	IEEE OUI Identifier (IEEE)
		TELHIDA1_LB	2	15:0	Telemetry Controller-Initiated Data Area 1 Last Block
		TELHIDA2_LB	2	31:16	Telemetry Controller-Initiated Data Area 2 Last Block
		TELHIDA3_LB	3	15:0	Telemetry Controller-Initiated Data Area 3 Last Block
		RSVD1	3	2959:16	Reserved (NVMe 1.3)
		RSVD1	3	2951:16	Reserved (NVMe 1.4)
		THI_DATA_GN	95	15:8	Telemetry Host-Initiated Data Generation Number (NVMe 1.4)
		THI_TELCI_DATA_AV	95	23:16	Telemetry Controller-Initiated Data Available
		THI_TELCI_DATA_GN	95	31:24	Telemetry Controller-Initiated Data Generation Number
		REASON_ID	96	1023:0	Reason Identifier
		THI_DATA_BLOCK	128	4096*n:0	Telemetry Host-Initiated Data Block
GetLogPage: Predictable Latency Per NVM Set (NVMe 1.4)	GetLogPage	PLP_NVM_SET_STATUS	0	7:0	Status
		RSVD	0	15:8	Reserved
		PLP_NVM_SET_EVENT_TYPE_BIT0	0	16	Event Type Bit 0
		PLP_NVM_SET_EVENT_TYPE_BIT1	0	17	Event Type Bit 1
		PLP_NVM_SET_EVENT_TYPE_BIT2	0	18	Event Type Bit 2
		RSVD1	0	29:19	Reserved
		PLP_NVM_SET_EVENT_TYPE_BIT14	0	30	Event Type Bit 14
		PLP_NVM_SET_EVENT_TYPE_BIT15	0	31	Event Type Bit 15
		RSVD2	1	223:0	Reserved
		PLP_NVM_SET_DTWIN_RT	8	63:0	DTWIN Reads Typical
		PLP_NVM_SET_DTWIN_WT	10	63:0	DTWIN Writes Typical
		PLP_NVM_SET_DTWIN_TM	12	63:0	DTWIN Time Maximum
		PLP_NVM_SET_NDWIN_TMH	14	63:0	NDWIN Time Minimum High
		PLP_NVM_SE	16	63:0	NDWIN Time Minimum Low

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
		T_NDWIN_TML			
		RSVD3	18	447:0	Reserved
		PLP_NVM_SET_DTWIN_RE	32	63:0	DTWIN Reads Estimate
		PLP_NVM_SET_DTWIN_WE	34	63:0	DTWIN Writes Estimate
		PLP_NVM_SET_DTWIN_TE	36	63:0	DTWIN Time Estimate
		RSVD4	38	2879:0	Reserved
GetLogPage: Telemetry Controller Initiated	GetLogPage	TCI_LOG_IDENTIFIER	0	7:0	Log Identifier
		RSVD	0	39:8	Reserved
		GLP_IEEE	1	31:8	IEEE OUI Identifier (IEEE)
		TELCIDA1_LB	2	15:0	Telemetry Controller-Initiated Data Area 1 Last Block
		TELCIDA2_LB	2	31:16	Telemetry Controller-Initiated Data Area 2 Last Block
		TELCIDA3_LB	3	15:0	Telemetry Controller-Initiated Data Area 3 Last Block
		RSVD1	3	2959:16	Reserved
		TCI_TELCI_DATA_AV	95	23:16	Telemetry Controller-Initiated Data Available
		TCI_TELCI_DATA_GN	95	31:24	Telemetry Controller-Initiated Data Generation Number
		REASON_ID	96	1023:0	Reason Identifier
		TCI_DATA_BLOCK	128	4096*n:0	Telemetry Controller-Initiated Data Block
GetLogPage: Persistent Event Log (NVMe 1.4)	GetLogPage	ACTION	10	9:8	Action
		RSVD	10	11:10	Reserved
GetLogPage: Endurance Group Information (NVMe 1.4)	GetLogPage	ENDURANCE_GROUP_CRITICAL_WARNING_BIT0	0	0	Critical Warning Bit 0
		RSVD	0	1	Reserved
		ENDURANCE_GROUP_CRITICAL_WARNING_BIT2	0	2	Critical Warning Bit 2
		ENDURANCE_GROUP_CRITICAL_WARNING_BIT3	0	3	Critical Warning Bit 3
		RSVD1	0	7:4	Reserved
		ENDURANCE_GROUP_LOG_AS	0	31:24	Available Spare

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
		ENDURANCE_GROUP_LOG_AS_TH	1	7:0	Available Spare Threshold
		ENDURANCE_GROUP_LOG_PER_USED	1	15:8	Percentage Used
		RSVD2	1	255:16	Reserved
		ENDURANCE_GROUP_LOG_EE	8	127:0	Endurance Estimate
		ENDURANCE_GROUP_LOG_DUR	12	127:0	Data Units Read
		ENDURANCE_GROUP_LOG_DUW	16	127:0	Data Units Written
		ENDURANCE_GROUP_LOG_MUW	20	127:0	Media Units Written
		ENDURANCE_GROUP_LOG_HRC	24	127:0	Host Read Commands
		ENDURANCE_GROUP_LOG_HWC	28	127:0	Host Write Commands
		ENDURANCE_GROUP_LOG_MDIE	32	127:0	Media and Data Integrity Errors
		ENDURANCE_GROUP_LOG_NEILE	36	127:0	Number of Error Information Log Entries
		RSVD3	40	2815:0	Reserved
GetLogPage: Sanitize Status	GetLogPage	SPROG	0	15:0	Sanitize Progress (SPROG)
		SSTAT	2	31:0	Sanitize Status
		SSTAT_BITS0_2	0	18:16	Sanitize Status Bits 2:0
		SSTAT_BITS3_7	0	23:19	Sanitize Status Bits 7:3
		SSTAT_BIT8	0	24:24	Sanitize Status Bit 8
		RSVD	0	31:25	Reserved
		SCDW10	1	31:0	Sanitize Command Dword 10 Information
		EST_TIME_FOW	2	31:0	Estimated Time For Overwrite
		EST_TIME_FBE	3	31:0	Estimated Time For Block Erase
		EST_TIME_FCE	4	31:0	Estimated Time For Crypto Erase

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
		RSVD1	5	3936:0	Reserved
GetLogPage: Persistent Event Log (Header) NVMe 1.4	GetLogPage	PEL_LID	0	7:0	Log Identifier
		RSVD	0	31:8	Reserved
		PEL_TNEV	1	31:0	Total Number of Events
		PEL_TLL	2	63:0	Total Log Length
		PEL_LOG_REV	4	7:0	Log Revision
		RSVD1	4	15:8	Reserved
		PEL_LOG_HDR_LEN	4	31:16	Log Header Length
		PEL_TIMESTAMP	5	63:0	Timestamp
		PEL_POW_ON_HOURS	7	127:0	Power on Hours
		PEL_POW_CYCLE_COUNT	11	63:0	Power Cycle Count
		VID	13	15:0	PCI Vendor ID
		SSVID	13	31:16	PCI Subsystem Vendor ID
		SN	14	159:0	Serial Number
		MN	19	319:0	Model Number
		SUBNQN	29	2047:0	NVM Subsystem NVMe Qualified Name
		RSVD2	93	863:0	Reserved
		RSVD3	120	0	Reserved
		PEL_SUP_EV_BITMAP_BIT_1	120	1	Smart / Health Log Snapshot Event Supported
		PEL_SUP_EV_BITMAP_BIT_2	120	2	Firmware Commit Event Supported
		PEL_SUP_EV_BITMAP_BIT_3	120	3	Timestamp Change Event Supported
		PEL_SUP_EV_BITMAP_BIT_4	120	4	Power-on or Reset Event Supported
		PEL_SUP_EV_BITMAP_BIT_5	120	5	NVM Subsystem Hardware Error Event Support
		PEL_SUP_EV_BITMAP_BIT_6	120	6	Change Namespace Event Supported
		PEL_SUP_EV_BITMAP_BIT_7	120	7	Format NVM Start Event Supported
		PEL_SUP_EV_BITMAP_BIT_8	120	8	Format NVM Completion Event Supported
		PEL_SUP_EV	120	9	Sanitize Start Event

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
		_BITMAP_BIT_9			Supported
		PEL_SUP_EV_BITMAP_BIT_10	120	10	Sanitize Completion Event Supported
		PEL_SUP_EV_BITMAP_BIT_11	120	11	Set Feature Event Supported
		PEL_SUP_EV_BITMAP_BIT_12	120	12	Telemetry Log Create Event Supported
		PEL_SUP_EV_BITMAP_BIT_13	120	13	Thermal Excursion Eevnt Supported
		RSVD4	120	221:14	Reserved
		PEL_SUP_EV_BITMAP_BIT_222	120	222	Vendor Specific Event Supported
		RSVD5	120	255:223	Reserved
GetLogPage: Persistent Event Header (NVMe 1.4)	GetLogPage	PEL_EVENT_TYPE	128	7:0	Event Type
		PEL_EVENT_TYPE_REV	128	15:8	Event Type Revision
		PEL_EVENT_HDR_LEN	128	23:16	Event Header Length
		RSVD6	128	31:24	Reserved
		PEL_CTRL_ID	129	15:0	Controller Identifier
		PEL_EVENT_TIMESTAMP	129	47:16	Event Timestamp
		RSVD7	130	63:16	Reserved
		PEL_VSIL	132	15:0	Vendor Specific Information Length
		PEL_EVENT_LEN	132	31:16	Event Length
GetLogPage: Persistent Event (Firmware Commit Event Data Format) (NVMe 1.4)	GetLogPage	PEL_FCE_OLD_FIRM_REV	133	63:0	Old Firmware Revision
		PEL_FCE_NEW_FIRM_REV	135	63:0	New Firmware Revision
		PEL_FCE_FIRM_COMMIT_ACTION	137	7:0	Firmware Commit Action
		PEL_FCE_FIRM_SLOT	137	15:8	Firmware Slot
		PEL_FCE_STATUS_CODE_TYPE	137	23:16	Status Code Type for Firmware Commit Command
		PEL_FCE_STATUS_RET	137	31:24	Status Returned for Firmware Commit Command

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
		PEL_FCE_VENDOR_ASSIGN_RET	138	15:0	Vendor Assigned Firmware Commit Result Code
GetLogPage: Persistent Event (Timestamp Change Event Data Format) (NVMe 1.4)	GetLogPage	PEL_TIMESTAMP_CHANGE_PREV	133	63:0	Previous Timestamp
		PEL_TIMESTAMP_CHANGE_MSEC_SINCE_RESET	135	63:0	Milliseconds Since Reset
GetLogPage: Persistent Event (Power-on or Reset Event Data Format) NVMe 1.4	GetLogPage	PEL_POW_ON_RES_FIRM_REV	133	63:0	Firmware Revision
		PEL_POW_ON_RES_RES_INFO_LIST	135	287:0	Reset Information List Element 0
		PEL_POW_ON_RES_RES_INFO_LIST	…	…	Reset Information List Element n - 1
GetLogPage: Reservation Notification	GetLogPage	LOG_PAGE_COUNT	0	63:0	Log Page Count
		RN_LOG_PAGE_TYPE	2	7:0	Reservation Notification Log Page Type
		NUMBER_OF_AVAILABLE_LOG_PAGES	2	15:8	Number of Available Log Pages
		RSVD	2	31:16	Reserved
		NAMESPACE_ID	3	31:0	Namespace ID
		RSVD1	4	415:0	Reserved
GetLogPage: Persistent Event (Sanitize Start Event Data Format) NVMe 1.4	GetLogPage	PEL_SANITIZE_START_SANICAP	133	31:0	SANICAP
		PEL_SANITIZE_START_SANITIZE_CDW10	134	31:0	Sanitize CDW10

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
		PEL_SANITIZE_START_SANITIZE_CDW11	135	31:0	Sanitize CDW11
GetLogPage: Persistent Event (Sanitize Completion Event Data Format) NVMe 1.4	GetLogPage	PEL_SANITIZE_COMPLETION_SANITIZE_PROGRESS	133	15:0	Sanitize Progress
		PEL_SANITIZE_COMPLETION_SANITIZE_STATUS	133	31:16	Sanitize Status
		PEL_SANITIZE_COMPLETION_CINFO	134	15:0	Completion Information
		RSVD	134	31:16	Reserved
GetLogPage: Persistent Event (Set Feature Event Data Format) NVMe 1.4	GetLogPage	PEL_SF_DWORD_COUNT	133	2:0	Dword Count
		PEL_SF_LOGGED_CMD_DW_0	133	3	Logged Command Completion Dword 0
		RSVD	133	15:4	Reserved
		PEL_SF_MEM_BUFF_COUNT	133	31:16	Memory Buffer Count
		PEL_SF_CMD_DWORDS	134	…	Command Dwords
		PEL_SF_CMD_MEM_BUFF	…	…	Memory Buffer

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
		PEL_SF_CMD_DW_0	…	…	Command Completion Dword 0
GetLogPage: Persistent Event (Format NVM Start Event Data Format) NVMe 1.4	GetLogPage	PEL_FORMAT_NVM_START_NS_IDENTIFIER	133	31:0	Namespace Identifier
		PEL_FORMAT_NVM_START_FNA	134	7:0	Format NVM Attributes
		RSVD	134	31:8	Reserved
		PEL_FORMAT_NVM_START_FORMAT_NVM_CDW10	135	31:0	Format NVM CDW10
GetLogPage: Persistent Event (Thermal Excursion Event Data Format) NVMe 1.4	GetLogPage	PEL_THERMAL_EXCURSION_OVER_TEMPERATURE	133	7:0	Over Temperature
		PEL_THERMAL_EXCURSION_THRESHOLD	133	15:8	Threshold
GetLogPage: Persistent Event (Format NVM Completion Event Data Format) NVMe 1.4	GetLogPage	PEL_FORMAT_NVM_COMPLETION_NS_IDENTIFIER	133	31:0	Namespace Identifier
		PEL_FORMAT_NVM_COMPLETION_SFPI	134	7:0	Smallest Format Progress Indicator

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
		PEL_FORMAT_NVM_COMPLETION_FORMAT_NVM_STATUS_ERROR	134	8	Format NVM Error
		PEL_FORMAT_NVM_COMPLETION_FORMAT_NVM_STATUS_INCF	134	9	Format NVM Incompatible Format
		RSVD	134	15:10	Reserved
		PEL_FORMAT_NVM_COMPLETION_CINFO	135	15:0	**Completion Information**
		PEL_FORMAT_NVM_COMPLETION_STATUS_FIELD	135	31:16	Status Field
GetLogPage: Persistent Event (Change Namespace Event Data Format) (NVMe 1.4)	GetLogPage	PEL_CHANGE_NAMESPACE_NS_CDW10	133	31:0	Namespace Management CDW10
		RSVD	134	31:0	Reserved
		PEL_CHANGE_NAMESPACE_NSZE	135	63:0	Namespace Size
		PEL_CHANGE_NAMESPACE_NCAP	137	127:0	Namespace Capacity
		PEL_CHANGE_NAMESPACE_FLBAS	141	7:0	Formatted LBA Size
		PEL_CHANGE_NAMESPACE_DPS	141	15:8	End-to-end Data Protection Type Settings
		PEL_CHANGE_NAMESPACE_NMIC	141	23:16	Namespace Multi-path I/O and Namespace Sharing Capabilities
		RSVD1	141	31:24	Reserved
		PEL_CHANGE_NAMESPACE_ANAGRID	142	31:0	ANA Group Identifier

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
		PEL_CHANGE_NAMESPACE_NVMSETID	143	15:0	NVM Set Identifier
		RSVD2	143	31:16	Reserved
		PEL_CHANGE_NAMESPACE_NSID	144	31:0	Namespace ID
GetLogPage: Predictable Latency Event Aggregate (NVMe 1.4)	GetLogPage_<index>	EG_EVENT_AGGR_NUM_ENTR	0	63:0	Number of Entries
		EG_EVENT_AGGR_ENTR	2	15:0	Entry 0
		EG_EVENT_AGGR_ENTR	2	31:16	Entry 1
		EG_EVENT_AGGR_ENTR	…	…	Entry n-1
GetLogPage: Asymmetric Namespace Access (NVMe 1.4)	GetLogPage_<index>	ANA_CHANGE_COUNT	0	63:0	Change Count
		ANA_NUM_GROUP_DESC	2	15:0	Number of ANA Group Descriptors
		RSVD	2	31:16	Reserved
		ANA_GROUP_ID	3	31:0	ANA Group ID
		ANA_NUM_NSID_VALUES	4	31:0	Number of NSID Values
		ANA_GROUP_DESC_CHANGE_COUNT	5	63:0	Change Count
		ANA_STATE	7	3:0	Asymmetric Namespace Access State
		RSVD1	7	7:4	Reserved
		RSVD2	7	255:8	Reserved
		ANA_NS_IDENTIFIER	...	...	Namespace Identifiers list
		…	…	…	Descriptor n - 1
GetLogPage: LBA Status Information (NVMe 1.4)	GetLogPage	LBA_LSLPLEN	0	31:0	LBA Status Log Page Length
		LBA_NLSLNE	1	31:0	Number of LBA Status Log Namespace Elements
		LBA_ESTULB	2	31:0	Estimate of Unrecoverable Logical Blocks
		RSVD	3	15:0	Reserved
		LBA_LSGC	3	31:16	LBA Status Generation Counter
		LBA_NEID	4	31:0	Namespace Element Identifier
		LBA_NLRD	5	31:0	Number of LBA Range

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
					Descriptors
		LBA_RATYPE	6	7:0	Recommended Action Type
		RSVD1	6	31:8	Reserved
		LBA_RANGE_DESC	7	127:0	LBA Range Descriptor 0
		LBA_RANGE_DESC	…	…	LBA Range Descriptor n - 1
GetLogPage: Endurance Group Event Aggregate (NVMe 1.4)	GetLogPage	EG_EVENT_AGGR_NUM_ENTR	0	63:0	Number of Entries
		EG_EVENT_AGGR_ENTR	2	15:0	Entry 0
		EG_EVENT_AGGR_ENTR	2	31:16	Entry 1
		EG_EVENT_AGGR_ENTR	…	…	Entry n - 1
Get Log Page: Changed Zone List (Zone Identifier List)	GetLogPage	CHANGED_ZN_LIST_NUMZID	0	15:0	Number of Zone Identifiers
		RSVD	0	63:16	Reserved
Get Log Page: Changed Zone List (Zone Identifier)	GetLogPage_ZNID<index>	CHANGED_ZN_LIST_ZID	0	63:0	Zone Identifier
Identify Primary Controller Capabilities	Identify	CNTLID	0	15:0	Controller ID
		PORTID	0	31:16	Port Identifier
		CRT_BIT0	1	0:0	Controller Resource Types bit 0
		CRT_BIT1	1	1:1	Controller Resource Types bit 1
		RSVD	1	7:2	Reserved
		RSVD1	1	223:8	Reserved
		VQFRT	8	31:0	VQ Resources Flexible Total
		VQRFA	9	31:0	VQ Resources Flexible Assigned
		VQRFAP	10	15:0	VQ Resources Flexible Allocated to Primary
		VQPRT	10	31:16	VQ Resources Private Total
		VQFRSM	11	15:0	VQ Resources Flexible Secondary Maximum
		VQGRAN	11	31:16	VQ Flexible Resource Preferred Granularity
		RSVD2	12	127:0	Reserved
		VIFRT	16	31:0	VI Resources Flexible Total
		VIRFA	17	31:0	VI Resources Flexible Assigned
		VIRFAP	18	15:0	VI Resources Flexible Allocated to Primary
		VIPRT	18	31:16	VI Resources Private Total
		VIFRSM	19	15:0	VI Resources Flexible Secondary Maximum
		VIGRAN	19	31:16	VI Flexible Resource

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
					Preferred Granularity
		RSVD3	20	32127 :0	Reserved
Identify Secondary Controller List	Identify	SCL_NID	0	7:0	Number of Identifiers
		RSVD	0	255:8	Reserved
Identify Secondary Controller List; Controller Entry	Identify_CE<index>	SCID	0	15:0	Secondary Controller Identifier
		PCID	0	31:16	Primary Controller Identifier
		SCS	4	15:0	Secondary Controller State
		Bit_0	1	0:0	Secondary Controller State bit 0
		RSVD	1	7:1	Reserved
		RSVD1	1	31:8	Reserved
		VFN	2	15:0	Virtual Function Number
		NVQ	2	31:16	Number of VQ Flexible Resources Assigned
		NVI	3	15:0	Number of VI Flexible Resources Assigned
		RSVD2	3	31:16	Reserved
Identify NSID List	Identify_ <index>	NIDT	0	7:0	Namespace Identifier Type
		NIDL	0	15:8	Namespace Identifier Length
		RSVD	0	31:16	Reserved
		NID	1	NIDL: 0	Namespace Identifier
Identify Controller; Power State Descriptor	Identify_ <index>	MP	0	15:0	Maximum Power
		RSVD11	0	23:16	Reserved
		MPS	0	24	Max Power Scale
		NOPS	0	25	Non-Operational State
		RSVD12	0	31:26	Reserved
		ENLAT	1	31:0	Entry Latency
		EXLAT	2	31:0	Exit Latency
		RRT	3	4:0	Relative Read Throughput
		RSVD13	3	7:5	Reserved
		RRL	3	12:8	Relative Read Latency
		RSVD14	3	15:13	Reserved
		RWT	3	20:16	Relative Write Throughput
		RSVD15	3	23:21	Reserved
		RWL	3	28:24	Relative Write Latency
		RSVD16	3	31:29	Reserved
		IDLP	4	15:0	Idle Power
		RSVD17	4	21:16	Reserved
		IPS	4	23:22	Idle Power Scale
		RSVD18	4	31:24	Reserved
		ACTP	5	15:0	Active Power
		APW	5	18:16	Active Power Workload
		RSVD19	5	21:19	Reserved
		APS	5	23:22	Active Power Scale
		RSVD20	5	95:24	Reserved

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
Identify Controller List	Identify_ <index>	LIST_CNTLID	0	15:0	Controller Identifier from Controller List
	Identify	CNTLID_COUNT	0	15:0	Controller Identifier Count
Identify Controller	Identify	VID	0	15:0	PCI Vendor ID
		SSVID	0	31:16	PCI Subsystem Vendor ID
		SN	1	159:0	Serial Number
		MN	6	319:0	Model Number
		FR	16	63:0	Firmware Revision
		RAB	18	7:0	Recommended Arbitration Burst
		IEEE	18	31:8	IEEE OUI Identifier
		CMIC_BIT0	19	0	Controller Multi-Path I/O and Namespace Sharing Capabilities Bit 0
		CMIC_BIT1	19	1	Controller Multi-Path I/O and Namespace Sharing Capabilities Bit 1
		CMIC_BIT2	19	2	Controller Multi-Path I/O and Namespace Sharing Capabilities Bit 2
		CMIC_BIT3	19	3	Controller Multi-Path I/O and Namespace Sharing Capabilities Bit 3 (NVMe 1.4)
		RSVD	19	7:3	Reserved
		RSVD	19	7:4	Reserved (NVMe 1.4)
		MDTS	19	15:8	Maximum Data Transfer Size
		CNTLID	19	31:16	Controller ID
		RSVD1	20	1407:0	Reserved
		VER	20	31:0	Version (NVMe 1.2)
		RTD3R	21	31:0	Runtime D3 Resume Latency (NVMe 1.2)
		RTD3E	22	31:0	Runtime D3 Entry Latency (NVMe 1.2)
		OAES	23	31:0	Optional Asynchronous Events Supported
		OAES_BIT8	23	8	Optional Asynchronous Events Supported Bit 8 (NVMe 1.2)
		OAES_BIT9	23	9	Optional Asynchronous Events Supported Bit 9 (NVMe 1.3)
		OAES_BIT10	23	10	Optional Asynchronous

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
					Events Supported Bit 10 (NVMe 1.4)
		OAES_BIT11	23	11	Optional Asynchronous Events Supported Bit 11 (NVMe 1.4)
		OAES_BIT12	23	12	Optional Asynchronous Events Supported Bit 12 (NVMe 1.4)
		OAES_BIT13	23	13	Optional Asynchronous Events Supported Bit 13 (NVMe 1.4)
		OAES_BIT14	23	14	Optional Asynchronous Events Supported Bit 14 (NVMe 1.4)
		RSVD2	23	31:15	Reserved (NVMe 1.4)
		CTRATT	24	31:0	Controller Attributes
		CTRATT_BIT 0	24	0	Controller Attributes, Bit 0
		CTRATT_BIT 1	24	1	Controller Attributes, Bit 1
		CTRATT_BIT 2	24	2	Controller Attributes, Bit 2 (NVMe 1.4)
		CTRATT_BIT 3	24	3	Controller Attributes, Bit 3 (NVMe 1.4)
		CTRATT_BIT 4	24	4	Controller Attributes, Bit 4 (NVMe 1.4)
		CTRATT_BIT 5	24	5	Controller Attributes, Bit 5 (NVMe 1.4)
		CTRATT_BIT 6	24	6	Controller Attributes, Bit 6 (NVMe 1.4)
		CTRATT_BIT 7	24	7	Controller Attributes, Bit 7 (NVMe 1.4)
		CTRATT_BIT 8	24	8	Controller Attributes, Bit 8 (NVMe 1.4)
		CTRATT_BIT 9	24	9	Controller Attributes, Bit 9 (NVMe 1.4)
		RSVD4	24	31:10	Reserved (NVMe 1.4)
		RRLS	25	15:0	Read Recovery Levels Supported (NVMe 1.4)
		RSVD3	25	87:16	Reserved (NVMe 1.4)
		CNTRLTYPE	27	31:24	Controller Type (NVMe 1.4)
		FGUID	28	127:0	FRU Globally Unique Identifier
		CRDT1	32	15:0	Command Retry Delay Time 1 (NVMe 1.4)
		CRDT2	32	31:16	Command Retry Delay Time 2 (NVMe 1.4)
		CRDT3	33	15:0	Command Retry Delay Time 3 (NVMe 1.4)
		RSVD5	33	855:16	Reserved (NVMe 1.4)

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
		OACS_BIT0	64	0	Optional Admin Command Support Bit 0
		OACS_BIT1	64	1	Optional Admin Command Support Bit 1
		OACS_BIT2	64	2	Optional Admin Command Support Bit 2
		OACS_BIT3	64	3	Optional Admin Command Support Bit 3 (NVMe 1.2)
		OACS_BIT4	64	4	Optional Admin Command Support Bit 4 (NVMe 1.3)
		OACS_BIT5	64	5	Optional Admin Command Support Bit 5 (NVMe 1.3)
		OACS_BIT6	64	6	Optional Admin Command Support Bit 6 (NVMe 1.3)
		OACS_BIT7	64	7	Optional Admin Command Support Bit 7 (NVMe 1.3)
		OACS_BIT8	64	8	Optional Admin Command Support Bit 8 (NVMe 1.3)
		OACS_BIT9	64	9	Optional Admin Command Support Bit 9 (NVMe 1.4)
		RSVD2	64	15:3	Reserved
		RSVD2	64	15:4	Reserved (NVMe 1.2)
		RSVD2	64	15:9	Reserved (NVMe 1.3)
		RSVD2	64	15:10	Reserved (NVMe 1.4)
		ACL	64	23:16	Abort Command Limit
		AERL	64	31:24	Asynchronous Event Request Limit
		FRMW_BIT0	65	0	Firmware Updates bit 0
		FRMW_BITS1_3	65	3:1	Firmware Updates bits 1:3
		FRMW_BIT4	65	4	Firmware Updates bit 4 (NVMe 1.2)
		RSVD3	65	7:4	Reserved
		RSVD3	65	7:5	Reserved (NVMe 1.2)
		LPA_BIT0	65	8	Log Page Attributes bit 0
		LPA_BIT1	65	9	Log Page Attributes bit 1 (NVMe 1.2)
		LPA_BIT2	65	10	Log Page Attributes bit 2 (NVMe 1.3)
		LPA_BIT3	65	11	Log Page Attributes bit 3 (NVMe 1.3)
		LPA_BIT4	65	12	Log Page Attributes bit 4 (NVMe 1.4)

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
		RSVD4	65	15:9	Reserved
		RSVD4	65	15:8	Reserved (NVMe 1.2)
		RSVD4	65	15:12	Reserved (NVMe 1.3)
		RSVD4	65	15:13	Reserved (NVMe 1.4)
		ELPE	65	23:16	Error Log Page Entries
		NPSS	65	31:24	Number of Power States Support
		AVSCC_BIT0	66	0	Admin Vendor Specific Command Configuration bit 0
		RSVD5	66	7:1	Reserved
		APSTA_BIT0	66	8	Autonomous Power State Transition Attributes bit 0
		RSVD4	66	15:9	Reserved
		RSVD5	66	1983:16	Reserved
		WCTEMP	66	31:16	Warning Composite Temperature Threshold (NVMe 1.2)
		CCTEMP	67	15:0	Critical Composite Temperature Threshold (NVMe 1.2)
		MTFA	67	31:16	Maximum Time for Firmware Activation (NVMe 1.2)
		HMPRE	68	31:0	Host Memory Buffer Preferred Size (NVMe 1.2)
		HMMIN	69	31:0	Host Memory Buffer Minimum Size (NVMe 1.2)
		TNVMCAP	70	127:0	Total NVM Capacity (NVMe 1.2)
		UNVMCAP	74	127:0	Unallocated NVM Capacity (NVMe 1.2)
		RPMBS_Unit Number	78	2:0	Number of RPMB Units (NVMe 1.2)
		RPMBS_Auth Method	78	5:3	Authentication Method (NVMe 1.2)
		RPMBS_Total Size	78	23:15	Total Size (NVMe 1.2)
		RPMBS_Acce ssSize	78	31:24	Access Size (NVMe 1.2)
		RSVD5	79	1568:0	Reserved (NVMe 1.2)
		EDSTT	79	15:0	Extended Device Self-test Time (NVMe 1.3)
		DSTO	79	23:16	Device Self-test Options (NVMe 1.3)
		FWUG	79	31:24	Firmware Update Granularity (NVMe 1.3)

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
		KAS	80	15:0	Keep Alive Support (NVMe 1.2.1)
		HCTMA_BIT0	80	16	Host Controlled Thermal Management Attributes bit 0 (NVMe 1.3)
		MNTMT	81	15:0	Minimum Thermal Management Temperature (NVMe 1.3)
		MXTMT	81	31:16	Maximum Thermal Management Temperature (NVMe 1.3)
		SANICAP_BIT0	82	0	Sanitize Capabilities bit 0 (NVMe 1.3)
		SANICAP_BIT1	82	1	Sanitize Capabilities bit 1 (NVMe 1.3)
		SANICAP_BIT2	82	2	Sanitize Capabilities bit 2 (NVMe 1.3)
		SANICAP_NDI	82	29	No-Deallocate Inhibited (NVMe 1.4)
		SANICAP_NODMMAS	82	31:30	No-Deallocate Modifies Media After Sanitize (NVMe 1.4)
		HMMINDS	83	31:0	Host Memory Buffer Minimum Descriptor Entry Size (NVMe 1.4)
		HMMAXD	84	15:0	Host Memory Maximum Descriptors Entries (NVMe 1.4)
		NSETIDMAX	84	31:16	NVM Set Identifier Maximum (NVMe 1.4)
		ENDGIDMAX	85	15:0	Endurance Group Identifier Maximum (NVMe 1.4)
		ANATT	85	23:16	ANA Transition Time (NVMe 1.4)
		ANACAP_BIT0	85	24	Asymmetric Namespace Access Capabilities Bit 0 (NVMe 1.4)
		ANACAP_BIT1	85	25	Asymmetric Namespace Access Capabilities Bit 1 (NVMe 1.4)
		ANACAP_BIT2	85	26	Asymmetric Namespace Access Capabilities Bit 2 (NVMe 1.4)
		ANACAP_BIT3	85	27	Asymmetric Namespace Access Capabilities Bit 3 (NVMe 1.4)
		ANACAP_BIT4	85	28	Asymmetric Namespace Access Capabilities Bit 4 (NVMe 1.4)
		ANACAP_BIT6	85	30	Asymmetric Namespace Access Capabilities Bit 6 (NVMe 1.4)
		ANACAP_BIT7	85	31	Asymmetric Namespace Access Capabilities Bit 7

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
					(NVMe 1.4)
		ANAGRPMAX	86	31:0	ANA Group Identifier Maximum (NVMe 1.4)
		NANAGRPID	87	31:0	Number of ANA Group Identifiers (NVMe 1.4)
		PELS	88	31:0	Persistent Event Log Size (NVMe 1.4)
		RSVD5	83	1439:0	Reserved (NVMe 1.3)
		RSVD5	89	1247:0	Reserved (NVMe 1.4)
		SQES_BITS0_3	128	3:0	Submission Queue Entry Size bits 3:0
		SQES_BITS4_7	128	7:4	Submission Queue Entry Size bits 7:4
		CQES_BITS0_3	128	11:8	Completion Queue Entry Size bits 3:0
		CQES_BITS4_7	128	15:12	Completion Queue Entry Size bits 7:4
		RSVD	128	31:16	Reserved
		MAXCMD	128	31:16	Maximum Outstanding Commands (NVMe 1.2.1)
		NN	129	31:0	Number of Namespaces
		ONCS_BIT0	130	0	Optional NVM Command Support bit 0
		ONCS_BIT1	130	1	Optional NVM Command Support bit 1
		ONCS_BIT2	130	2	Optional NVM Command Support bit 2
		ONCS_BIT3	130	3	Optional NVM Command Support bit 3
		ONCS_BIT4	130	4	Optional NVM Command Support bit 4
		ONCS_BIT5	130	5	Optional NVM Command Support bit 5
		ONCS_BIT6	130	6	Optional NVM Command Support bit 6 (NVMe 1.3)
		ONCS_BIT7	130	7	Optional NVM Command Support bit 6 (NVMe 1.4)
		RSVD1	130	15:8	Reserved
		FUSES_BIT0	130	16	Fused Operation Support bit 0
		RSVD2	130	31:17	Reserved
		FNA_BIT0	131	0	Format NVM Attributes bit 0
		FNA_BIT1	131	1	Format NVM Attributes bit 1
		FNA_BIT2	131	2	Format NVM Attributes bit 2

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
		RSVD3	131	7:3	Reserved
		VWC_BIT0	131	8	Volatile Write Cache bit 0
		VWC_BITS12	131	10:9	Volatile Write Cache bits 1-2 (NVMe 1.4)
		RSVD4	131	15:11	Reserved
		AWUN	131	31:16	Atomic Write Unit Normal
		AWUPF	132	15:0	Atomic Write Unit Power Fail
		NVSCC_BIT0	132	16	NVM Vendor Specific Command Configuration bit 0
		RSVD5	132	23:17	Reserved
		NWPC_BIT0	132	24	Namespace Write Protection Capabilities bit 0 (NVMe 1.4)
		NWPC_BIT1	132	25	Namespace Write Protection Capabilities bit 1 (NVMe 1.4)
		NWPC_BIT2	132	26	Namespace Write Protection Capabilities bit 2 (NVMe 1.4)
		RSVD6	132	31:24	Reserved
		RSVD6	132	31:27	Reserved (NVMe 1.4)
		ACWU	133	15:0	Atomic Compare & Write Unit
		RSVD	133	31:16	Reserved
		SGL_SUPPORT_BIT0, SGLS_BIT0	134	0	SGL Support bit 0
		SGL_SUPPORT_BIT01	134	1:0	SGL Support bits 0-1
		SGL_SUPPORT_BIT2, SGLS_BIT2	134	2	SGL Support bit 2
		RSVD7	134	15:1	Reserved
		SGL_SUPPORT_BIT16, SGLS_BIT16	134	16	SGL Support bit 16
		SGL_SUPPORT_BIT17, SGLS_BIT17	134	17	SGL Support bit 17 (NVMe 1.2)
		SGL_SUPPORT_BIT18, SGLS_BIT18	134	18	SGL Support bit 18 (NVMe 1.2)
		SGL_SUPPORT_BIT19	134	19	SGL Support bit 19 (NVMe 1.2.1)
		SGL_SUPPORT_BIT20	134	20	SGL Support bit 20 (NVMe 1.2.1)
		RSVD8	134	31:17	Reserved
		RSVD8	134	31:19	Reserved (NVMe 1.2)
		MNAN	135	31:0	Maximum Number of Allowed Namespaces (NVMe 1.4)

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
		RSVD9	135	1311:0	Reserved
		RSVD9	136	1307:0	Reserved (NVMe 1.4)
		RSVD10	176	10751:0	Reserved
		RSVD9	135	1823:0	Reserved (NVMe 1.2.1)
		SUBNQN	192	1019:0	NVM Subsystem NVMe Qualified Name
		RSVD10	256	8191:0	Reserved (NVMe 1.2.1)
		VS	768	8191:0	Vendor Specific
Identify Namespace; LBA Format	Identify_LBAF<index>	MS	0	15:0	Metadata Size
		LBADS	0	23:16	LBA Data Size
		RP	0	25:24	Relative Performance
		RSVD8	0	31:26	Reserved
Identify Namespace	Identify	NSZE	0	63:0	Namespace Size
		NCAP	2	63:0	Namespace Capacity
		NUSE	4	63:0	Namespace Utilization
		NSFEAT_BIT0	6	0	Namespace Features bit 0
		NSFEAT_BIT1	6	1	Namespace Features bit 1 (NVMe 1.2)
		NSFEAT_BIT2	6	2	Namespace Features bit 2 (NVMe 1.2)
		NSFEAT_BIT3	6	3	Namespace Features bit 3 (NVMe 1.3)
		NSFEAT_BIT4	6	4	Namespace Features bit 4 (NVMe 1.4)
		RSVD	6	7:1	Reserved
		RSVD	6	7:3	Reserved (NVMe 1.2)
		RSVD	6	7:4	Reserved (NVMe 1.3)
		RSVD	6	7:5	Reserved (NVMe 1.4)
		NLBAF	6	15:8	Number of LBA Formats
		FLBAS_BITS0_3	6	19:16	Formatted LBA Size bits 0 3
		FLBAS_BIT4	6	20	Formatted LBA Size bit 4
		RSVD1	6	23:21	Reserved
		MC_BIT0	6	24	Metadata Capabilities bit 0
		MC_BIT1	6	25	Metadata Capabilities bit 1
		RSVD2	6	31:26	Reserved
		DPC_BIT0	7	0	End-to-end Data Protection Capabilities bit 0
		DPC_BIT1	7	1	End-to-end Data Protection Capabilities bit 1

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
		DPC_BIT2	7	2	End-to-end Data Protection Capabilities bit 2
		DPC_BIT3	7	3	End-to-end Data Protection Capabilities bit 3
		DPC_BIT4	7	4	End-to-end Data Protection Capabilities bit 4
		RSVD3	7	7:5	Reserved
		DPS_BITS0_2	7	10:8	End-to-end Data Protection Type Settings bits 0 2
		DPS_BIT3	7	11	End-to-end Data Protection Type Settings bit 3
		RSVD4	7	15:12	Reserved
		NMIC_BIT0	7	16	Namespace Multi-path I/O and Namespace Sharing Capabilities bit 0
		RSVD5	7	23:17	Reserved
		RESCAP_BIT 0	7	24	Reservation Capabilities bit 0
		RESCAP_BIT 1	7	25	Reservation Capabilities bit 1
		RESCAP_BIT 2	7	26	Reservation Capabilities bit 2
		RESCAP_BIT 3	7	27	Reservation Capabilities bit 3
		RESCAP_BIT 4	7	28	Reservation Capabilities bit 4
		RESCAP_BIT 5	7	29	Reservation Capabilities bit 5
		RESCAP_BIT 6	7	30	Reservation Capabilities bit 6
		RESCAP_BIT 7	7	31	Reservation Capabilities bit 7 (NVMe 1.3)
		RSVD6	7	31	Reserved
		RSVD7	8	703:0	Reserved
		FPI	8	15:0	Format Progress Indicator
		FPI_BITS0_6	8	6:0	Format Progress Indicator, bits 0:6 (NVMe 1.2)
		FPI_BIT7	8	7	Format Progress Indicator, bit 7 (NVMe 1.2)
		RSVD7	8	15:8	Reserved (NVMe 1.2)
		DLFEAT_BIT0 2	8	10:8	Deallocate Logical Block Features bits 0-2 (NVMe 1.3)
		DLFEAT_BIT3	8	11	Deallocate Logical Block Features bit 3 (NVMe 1.3)
		DLFEAT_BIT4	8	12	Deallocate Logical Block Features bit 4 (NVMe 1.3)
		NAWUN	8	31:16	Namespace Atomic Write Unit Normal (NVMe 1.2)

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
		NAWUPF	9	15:0	Namespace Atomic Write Unit Power Fail (NVMe 1.2)
		NACWU	9	31:16	Namespace Atomic Compare & Write Unit (NVMe 1.2)
		NABSN	10	15:0	Namespace Atomic Boundary Size Normal (NVMe 1.2)
		NABO	10	31:16	Namespace Atomic Boundary Offset (NVMe 1.2)
		NABSPF	11	15:0	Namespace Atomic Boundary Size Power Fail (NVMe 1.2)
		RSVD11	11	31:16	Reserved (NVMe 1.2)
		NOIOB	11	31:16	Namespace Optimal IO Boundary (NVMe 1.3)
		NVMCAP	12	127:0	NVM Capacity (NVMe 1.2)
		RSVD11	16	319:0	Reserved (NVMe 1.2)
		NPWG	16	15:0	Namespace Preferred Write Granularity (NVMe 1.4)
		NPWA	16	31:16	Namespace Preferred Write Alignment (NVMe 1.4)
		NPDG	17	15:0	Namespace Preferred Deallocate Granularity (NVMe 1.4)
		NPDA	17	31:16	Namespace Preferred Deallocate Alignment (NVMe 1.4)
		NOWS	18	15:0	Namespace Optimal Write Size (NVMe 1.4)
		RSVD11	18	159:16	Reserved (NVMe 1.4)
		ANAGRPID	23	31:0	ANA Group Identifier (NVMe 1.4)
		RSVD12	24	23:0	Reserved (NVMe 1.4)
		NSATTR	24	31:24	Namespace Attributes (NVMe 1.4)
		NVMSETID	25	15:0	NVM Set Identifier (NVMe 1.4)
		ENDGID	25	31:16	Endurance Group Identifier (NVMe 1.4)
		NGUID	26	127:0	Namespace Globally Unique Identifier (NVMe 1.2)
		EUI64	30	63:0	IEEE Extended Unique Identifier
		RSVD9	48	1535:0	Reserved
		VS	96	29695:0	Vendor Specific
Namespace Management, Create	NamespaceManagement	NSZE	0	63:0	Namespace Size
		NCAP	2	63:0	Namespace Capacity
		RSVD	4	79:0	Reserved

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
		FLBAS_BITS0_3	6	19:16	Formatted LBA Size bits 0 3
		FLBAS_BIT4	6	20	Formatted LBA Size bit 4
		RSVD1	6	23:21	Reserved
		RSVD2	6	39:26	Reserved
		DPS_BITS0_2	7	10:8	End-to-end Data Protection Type Settings bits 0 2
		DPS_BIT3	7	11	End-to-end Data Protection Type Settings bit 3
		RSVD3	7	7:5	Reserved
		NMIC_BIT0	7	16	Namespace Multi-path I/O and Namespace Sharing Capabilities bit 0
		RSVD4	7	23:17	Reserved
		RSVD5	7	487:24	Reserved
		ANAGRPID	23	31:0	ANA Group Identifier
		RSVD6	24	31:0	Reserved (NVMe 1.4)
		NVMSETID	25	15:0	NVM Set Identifier (NVMe 1.4)
		RSVD7	25	2255:16	Reserved (NVMe 1.4)
Identify: List of Namespaces, NSID, I/O Command Set Specific Active/Allocated NSID list Get Log Page, Changed Namespace List	Identify_NSID<index>, GetLogPage_NSID<index>	NSID	0	31:0	Namespace ID
Identify NVM Set List	Identify	NVMSL_NID	0	7:0	Number of Identifiers (NVMe 1.4)
		RSVD	0	511:8	Reserved (NVMe 1.4)
Identify NVM Set List; Attributes Entry	Identify_AE<index>	NVMSI	0	15:0	NVM Set Identifier (NVMe 1.4)
		EGI	0	31:16	Endurance Group Identifier (NVMe 1.4)
		RSVD_TABLE	1	31:0	Reserved (NVMe 1.4)
		R4KiBRT	2	31:0	Random 4 KiB Read Typical (NVMe 1.4)
		OWS	3	31:0	Optimal Write Size (NVMe 1.4)

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
		TNVMSC	4	127:0	Total NVM Set Capacity (NVMe 1.4)
		UANVMSC	8	127:0	Unallocated NVM Set Capacity (NVMe 1.4)
		RSVD_TABLE 1	12	639:0	Reserved (NVMe 1.4)
Identify Namespace Granularity List	Identify	NGA_GDM	0	0	Namespace Granularity Attributes: Granularity Descriptor Mapping (NVMe 1.4)
		NGA_RSVD	0	31:1	Reserved (NVMe 1.4)
		NDS	1	7:0	Number of Descriptors (NVMe 1.4)
		RSVD	1	207:8	Reserved (NVMe 1.4)
Identify Namespace Granularity List; Namespace Granularity Descriptor	Identify_NGD<index>	NSG	0	63:0	Namespace Size Granularity (NVMe 1.4)
		NCG	2	63:0	Namespace Capacity Granularity (NVMe 1.4)
Identify UUID List	Identify	RSVD	0	255:0	Reserved (NVMe 1.4)
Identify UUID List; UUID List Entry	Identify_ULE<index>	UUIDLEH_IDA	0	1:0	UUID Lists Entry Header: Identifier Association (NVMe 1.4)
		RSVD_TABLE	0	7:2	Reserved (NVMe 1.4)
		RSVD_TABLE 1	0	127:8	Reserved (NVMe 1.4)
		UUID	4	127:0	UUID (NVMe 1.4)
Identify, I/O Command Set Vector	Identify_IOCSV<index>	NVM_COM_SET	0	0	NVM Command Set
		RSVD	0	1	Reserved
		ZN_COM_SET		2	Zoned Namespace Command Set
		RSVD1	0	63:3	Reserved
Identify, Zoned Namespace Command Set Identify Namespace	Identify	ZOC_BIT0	0	0	Zone Operation Characteristics: Variable Zone Capacity
		ZOC_BIT1	0	1	Zone Operation Characteristics: Zone Active Excursions
		RSVD	0	15:2	Reserved

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
		OZCS_BIT0	0	16	Optional Zoned Command Support: Read Across Zone Boundaries
		RSVD1	0	31:17	Reserved
		MAR	1	31:0	Maximum Active Resources
		MOR	2	31:0	Maximum Open Resources
		RLL	3	31:0	Reset Recommended Limit
		FRL	4	31:0	Finish Recommended Limit
		RSVD2	5	22367:0	Reserved
		RSVD3	768	6144:0	Reserved
		VS	960	2048:0	Vendor Specific
Identify, Zoned Namespace Command Set Identify Namespace, LBA Format Extension	Identify_LBAFE<index>	ZS	0	63:0	Zone Size
		ZDES	2	7:0	Zone Descriptor Extension Size
		RSVD	2	63:8	Reserved
Identify, Zoned Namespace Command Set Identify Controller	Identify	ZASL	0	7:0	Zone Append Size Limit
		RSVD	0	32767:8	Reserved
Namespace Attachment	NamespaceAttachment_CNTLID<index>	LIST_CNTLID	0	15:0	Controller Identifier from Controller List
	NamespaceAttachment	CNTLID_COUNT	0	15:0	Controller Identifier Count
Reservation Acquire	ReservationAcquire	CRKEY	0	63:0	Current Reservation Key
		PRKEY	2	63:0	Preempt Reservation Key
Reservation Register	ReservationRegister	CRKEY	0	63:0	Current Reservation Key
		NRKEY	2	63:0	New Reservation Key
Reservation Release	ReservationRelease	CRKEY	0	63:0	Current Reservation Key
Reservation Report	ReservationReport	GEN	0	31:0	Generation
		RTYPE	1	7:0	Reservation Type
		REGCTL	1	23:8	Number of Registered Controllers
		RSVD	1	39:24	Reserved
		PTPLS	2	15:8	Persist Through Power Loss State
		RSVD1	2	127:1	Reserved

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
				6	
Reservation Report	ReservationReport	RCSTS	2	15:0	Reservation Status
Reservation Report; Registered Controller	ReservationReport_SD<index>	CNTLID	0	15:0	Controller ID
		RCSTS_BIT0	0	16	Reservation Status bit 0
		RSVD	0	23:17	Reserved
		RSVD1	0	63:24	Reserved
		HOSTID	2	63:0	Host Identifier
		RKEY	4	63:0	Reservation Key
		RKEY	2	63:0	Reservation Key (Extended)
		HOSTID	4	127:0	Host Identifier (Extended)
		RSVD2	8	255:0	Reserved (Extended)
DatasetManagement: Range Definition	DatasetManagement <index>	AF	0	3:0	Access Frequency
		AL	0	5:4	Access Latency
		RSVD2	0	7:6	Reserved
		SR	0	8	Sequential Read Range
		SW	0	9	Sequential Write Range
		WP	0	10	Write Prepare
		RSVD3	0	23:11	Reserved
		CAS	0	31:24	Command Access Size
		LEN_LB	1	31:0	Length in logical blocks
		SLBA	2	63:0	Starting LBA
Zone Management Receive: Report Zones Data Structure, Extended Report Zones Data Structure	ZoneManagementReceive	NUMZ	0	7:0	Number of Zones
		RSVD	0	63:8	Reserved
Zone Management Receive: Zone Descriptor	ZoneManagementReceive_ZND<index>	ZT	0	3:0	Zone Type
		RSVD_TABLE	0	7:4	Reserved
		RSVD_TABLE 1	0	11:8	Reserved
		ZS	0	15:12	Zone State
		ZFC	0	16	Zone Finished by Controller
		FZR	0	17	Finish Zone Recommended
		RZR	0	18	Reset Zone Recommended
		RSVD_TABLE 2	0	22:19	Reserved
		ZDEV	0	23	Zone Descriptor Extension Valid
		RSVD_TABLE 3	0	63:24	Reserved
		ZCAP	2	63:0	Zone Capacity
		ZSLBA	4	63:0	Zone Start LBA

Command	Parameter	Fields	starting DWORD	Offset in bits	Meaning
		WP	6	63:0	Write Pointer
		RSVD_TABLE4	8	1023:0	Reserved
Zone Management Receive: Zone Descriptor Extension	ZoneManagementReceive_ZND<index>	ZDE	0	ZDES*64*8:0	Zone Descriptor Extension

The Feature Identifier field of Get Feature and Set Feature commands can be accessed the following way: **in.GetFeature_FID** and **in.SetFeature_FID** respectively. The following possible values are defined by VSE and the corresponding constants can be used by scripts:

Feature Identifier	Value	Name
_NVMC_FID_ARBITRATION	0x01	Arbitration
_NVMC_FID_POWERMANAGEMENT	0x02	Power Management
_NVMC_FID_LBARANGETYPE	0x03	LBA Range Type
_NVMC_FID_TEMPERATURETHRESHOLD	0x04	Temperature Threshold
_NVMC_FID_ERRORRECOVERY	0x05	Error Recovery
_NVMC_FID_VOLATILEWRITECACHE	0x06	Volatile Write Cache
_NVMC_FID_NUMBEROFQUEUES	0x07	Number of Queues
_NVMC_FID_INTERRUPTCOALESCING	0x08	Interrupt Coalescing
_NVMC_FID_INTERRUPTVECTORCONFIG	0x09	Interrupt Vector Configuration
_NVMC_FID_WRITEATOMICITY	0x0A	Write Atomicity
_NVMC_FID_ASYNCEVENTCONFIG	0x0B	Asynchronous Event Configuration
_NVMC_FID_AUTOPOWERSTATETRANS	0x0C	Autonomous Power State Transition
_NVMC_FID_HOSTMEMORYBUFFER	0x0D	Host Memory Buffer
_NVMC_FID_TIMESTAMP	0x0E	Timestamp
_NVMC_FID_KEEPALIVETIMER	0x0F	Keep Alive Timer
_NVMC_FID_HOST_CONTROLLED_THERMAL_MANAGEMENT	0x10	Host Controlled Thermal Management
_NVMC_FID_NON_OPERATIONAL_POWER_STATE_CONFIG	0x11	Non Operational Power State Config
_NVMC_FID_READ_RECOVERY_LEVEL_CONFIG	0x12	FID_READ_RECOVERY_LEVEL_CONFIG
_NVMC_FID_PREDICTABLE_LATENCY_MODE_CONFIG	0x13	FID_PREDICTABLE_LATENCY_MODE_CONFIG
_NVMC_FID_PREDICTABLE_LATENCY_MODE_WINDOW	0x14	FID_PREDICTABLE_LATENCY_MODE_WINDOW
_NVMC_FID_LBA_STATUS_INF_ATTRIBUTES	0x15	FID_LBA_STATUS_INF_ATTRIBUTES
_NVMC_FID_HOST_BEHAVIOR_SUPPORT	0x16	FID_HOST_BEHAVIOR_SUPPORT
_NVMC_FID_SANITIZE_CONFIG	0x17	FID_SANITIZE_CONFIG
_NVMC_FID_ENDURANCE_GROUP_EVENT_CONFIG	0x18	FID_ENDURANCE_GROUP_EVENT_CONFIG
_NVMC_FID_IO_COMMAND_SET_PROFILE	0x19	I/O Command Set Profile
_NVMC_FID_SOFTPROGRESMARKER	0x80	Software Progress Marker
_NVMC_FID_HOSTIDENTIFIER	0x81	Host Identifier
_NVMC_FID_RESERVNOTIFICMASK	0x82	Reservation Notification Mask
_NVMC_FID_RESERVPERSISTANCE	0x83	Reservation Persistance
_NVMC_FID_NAMESPACE_WRITE_PROTECT_CONFIG	0x84	FID_NAMESPACE_WRITE_PROTECT_CONFIG

The Log Page Identifier field of Get Log Page command can be accessed the following way: **in.GetLogPage_LID**. The following possible values are defined by VSE and the corresponding constants can be used by scripts:

Log Page Identifier	Value	Name
_NVMC_LID_ERROR_INFORMATION	0x01	Error Information
_NVMC_LID_SMART_HEALTH_INFORMATION	0x02	SMART / Health Information
_NVMC_LID_FIRMWARE_SLOT_INFORMATION	0x03	Firmware Slot Information
_NVMC_LID_CHANGED_NAMESPACE_LIST	0x04	Changed Namespace List
_NVMC_LID_COMMAND_EFFECTS_LOG	0x05	Command Effects Log
_NVMC_LID_DEVICE_SELF_TEST	0x06	Device Self-test
_NVMC_LID_TELEMETRY_HOST_INITIATED	0x07	Telemetry Host-Initiated
_NVMC_LID_TELEMETRY_CONTROLLER_INITIATED	0x08	Telemetry Controller-Initiated
_NVMC_LID_PERSISTENT_EVENT_LOG	0x0D	Persistent Event Log
_NVMC_LID_RESERVATION_NOTIFICATION	0x80	Reservation Notification
_NVMC_LID_SANITIZE_STATUS	0x81	Sanitize Status
_NVMC_LID_CHANGED_ZONE_LIST	0xBF	Changed Zone List

The Controller or Namespace Structure of Identify command can be accessed the following way: **in.Identify_CNS**. The following possible values are defined by VSE and the corresponding constants can be used by scripts:

Controller or Namespace structure	Value	Name
_NVMC_IDENTIFY_CNS_NAMESPACE	0x00	The Identify Namespace structure if the namespace is attached to this controller (otherwise structure is zero-filled)
_NVMC_IDENTIFY_CNS_CONTROLLER	0x01	The Identify Controller
_NVMC_IDENTIFY_CNS_LIST_OF_NAMESPACES	0x02	A list of up to 1024 namespace Ids attached to this controller
_NVMC_IDENTIFY_CNS_LIST_OF_NSID_STRUCTURES	0x03	List of Namespace Identification Descriptor structures
_NVMC_IDENTIFY_CNS_NVM_SET_LIST	0x04	An NVM Set List is returned to the host for up to 31 NVM Sets. The list contains entries for NVM Set identifiers greater than or equal to the value specified in the NVM Set Identifier field.
_NVMC_IDENTIFY_CNS_IO_COMMAND_SET_NS	0x05	Identify I/O Command Set Specific Namespace data structure for the specified NSID for the I/O Command Set specified in the CSI field.
_NVMC_IDENTIFY_CNS_IO_COMMAND_SET_CONTROLLER	0x06	Identify I/O Command Set Specific Controller data structure for the controller processing the command.
_NVMC_IDENTIFY_CNS_IO_COMMAND_SET_ACTIVE_NS_ID	0x07	Active Namespace ID list associated with the specified I/O Command Set.
_NVMC_IDENTIFY_CNS_RSVD	0x08	Reserved
_NVMC_IDENTIFY_CNS_LIST_OF_NAMESPACES_ALL	0x10	A list of up to 1024 namespace Ids present in NVM subsystem with values greater than CDW1.NSID.
_NVMC_IDENTIFY_CNS_NAMESPACE_ALL	0x11	The Identify Namespace structure. Namespace may or may not be attached to the controller.
_NVMC_IDENTIFY_CNS_LIST_OF_CONTROLLERS	0x12	The Controller List structure for controllers attached to namespace specified in CDW1.NSID.
_NVMC_IDENTIFY_CNS_LIST_OF_CONTROLLERS_ATTACHED	0x13	The controller List structure. They may or may not be attached to namespaces, with values greater than CDW10.CNTID.
_NVMC_IDENTIFY_CNS_PRIMARY_CONTOLLER_CAP	0x14	The Primary Controller Capabilities Structure for the primary controller

		specified.
_NVMC_IDENTIFY_CNS_SECONDARY_CONTROLLER_LIST	0x15	A Secondary Controller List t. Up to 127 secondary controllers associated with the primary controller issuing this command.
_NVMC_IDENTIFY_CNS_NAMESPACE_GRANULARITY_LIST	0x16	A Namespace Granularity List is returned to the host for up to sixteen Namespace Granularity Entries.
_NVMC_IDENTIFY_CNS_UUID_LIST	0x17	A UUID List is returned to the host.
_NVMC_IDENTIFY_CNS_IO_COMMAND_SET_ALLOCATED_NS_ID	0x1A	I/O Command Set Specific Allocated Namespace ID list.
_NVMC_IDENTIFY_CNS_IO_COMMAND_SET_NS_FOR_ALLOCATED_NS_ID	0x1B	I/O Command Set Specific Identify Namespace data structure for an Allocated Namespace ID.
_NVMC_IDENTIFY_CNS_IDENTIFY_IO_COMMAND_SET	0x1C	I/O Command Set data structure.

The Asynchronous Event Type of Asynchronous Event Request command can be accessed the following way: **in. AsyncEventRequest_AE_TYPE**. The following possible values are defined by VSE and the corresponding constants can be used by scripts:

Asynchronous Event Type	Value	Name
_NVMC_ASYNC_EVENT_TYPE_ERROR_STATUS	0x00	Error Status
_NVMC_ASYNC_EVENT_TYPE_SMART_HEALTH_STATUS	0x01	SMART / Health Status
_NVMC_ASYNC_EVENT_TYPE_RSVD_FIRST	0x02	Reserved
_NVMC_ASYNC_EVENT_TYPE_RSVD_LAST	0x05	Reserved
_NVMC_ASYNC_EVENT_TYPE_ID_COMMAND_SET_SPECIFIC_STATUS	0x06	I/O Command Set Specific status
_NVMC_ASYNC_EVENT_TYPE_VENDOR_SPECIFIC	0x07	Vendor Specific

5.2.13.1 Metric values

in.Metric_Throughput: Metric presenting transaction payload divided by response time, expressed in **kilobytes** per second, an integer value

in.Metric_PayloadBytes: Metric presenting number of data payload bytes this NVM Transaction transferred, an integer value.

in.Metric_NumOfNVMTras: Metric presenting the total number of NVM Transactions that compose this NVM Command, an integer value.

in. Metric_LatencyTime: Metric presenting time measured from the end of transmission of the SQ Doorbell to the completion of data delivery, a VSE time object value (see 9.1 VSE Time Object for details)

in. Metric_ResponseTime: Metric presenting time it took to transmit this NVM Command on the link, from the beginning of the first packet to the end of the last packet in the command, a VSE time object value (see 9.1 VSE Time Object for details)

in. Metric_SubmissionDoorbell_CompletionDoorbell_DeltaTime: Metric presenting time measured between Submission Doorbell transaction and Completion Doorbell transaction, a VSE time object value (see 9.1 VSE Time Object for details)

in. Metric_SubmissionDoorbell_CompletionCommand_DeltaTime: Metric presenting time measured between Submission Doorbell transaction and Completion Command transaction, a VSE time object value (see 9.1 VSE Time Object for details)

in. Metric_SubmissionCommand_CompletionCommand_DeltaTime: Metric presenting time measured between Submission Command transaction and Completion Command transaction, a VSE time object value (see 9.1 VSE Time Object for details)

5.2.14 Framing Token specific set of members

in.FramingTokenType: Contains the numeric encoding of the Framing Token type. The following possible values are defined by VSE and the corresponding constants can be used by scripts:

```
FT_TYPE_EDB = 0x0D;
FT_TYPE_EDS = 0x0E;
```

5.3 AHCI transaction-specific set of members

Valid for AHCI transactions only. Undefined for other events.

All the AHCI-specific values are present in the input context for AHCI transactions, depending upon the type of register. Also the common PayloadLength and Payload values reflect the total combined payload for the AHCI transaction. In addition to that, the following values exist:

in.AHCIRegId: Contains the numeric encoding of the AHCI register type. The following possible values are defined by VSE and the corresponding constants can be used by scripts:

Register type	Value	Register description	Corresponding event type
_AHCI_CAP	2	Host Capabilities	_AHCI_HBA_REG
_AHCI_GHC	3	Global Host Control	
_AHCI_IS	4	Interrupt Status	
_AHCI_PI	5	Ports Implemented	
_AHCI_VS	6	Version	
_AHCI_CCC_CTL	7	Command Completion Coalescing Control	
_AHCI_CCC_PORTS	8	Command Completion Coalescing Ports	
_AHCI_EM_LOC	9	Enclosure Management Location	
_AHCI_EM_CTL	10	Enclosure Management Control	
_AHCI_CAP2	11	Host Capabilities Extended	
_AHCI_BOHC	12	BIOS/OS Handoff Control and Status	
_AHCI_RESERVED_HBA	13	Reserved	
_AHCI_RESERVED_NVMHCI	14	Reserved for NVMHCI	
_AHCI_VENDOR_SPECIFIC_REGISTERS	15	Vendor Specific	
_AHCI_PxCLB	16	Port x Command List Base Address	_AHCI_PORT_REG
_AHCI_PxCLBU	17	Port x Command List Base Address Upper 32-Bits	

_AHCI_PxFB	18	Port x FIS Base Address	
_AHCI_PxFBU	19	Port x FIS Base Address Upper 32-Bits	
_AHCI_PxIS	20	Port x Interrupt Status	
_AHCI_PxIE	21	Port x Interrupt Enable	
_AHCI_PxCMD	22	Port x Command and Status	
_AHCI_PxReserved1	23	Reserved1	
_AHCI_PxTFD	24	Port x Task File Data	
_AHCI_PxSIG	25	Port x Signature	
_AHCI_PxSSTS	26	Port x Serial ATA Status (SCR0: SStatus)	
_AHCI_PxSCTL	27	Port x Serial ATA Control (SCR2: SControl)	
_AHCI_PxSERR	28	Port x Serial ATA Error (SCR1: SError)	
_AHCI_PxSACT	29	Port x Serial ATA Active (SCR3: SActive)	
_AHCI_PxCI	30	Port x Command Issue	
_AHCI_PxSNTF	31	Port x Serial ATA Notification (SCR4: SNotification)	
_AHCI_PxFBS	32	Port x FIS-based Switching Control	
_AHCI_PxDEVSLP	33	Port x Device Sleep	
_AHCI_PxReserved2	34	Reserved2	
_AHCI_PxVS	35	Port x Vendor Specific	
_AHCI_COMMAND_HEADER	36	Command header	_AHCI_CMND_LIST
_AHCI_DSFIS	44	DMA Setup FIS	_AHCI_RECEIVED_FIS
_AHCI_PSFIS	45	PIO Setup FIS	
_AHCI_RFIS	46	D2H Register FIS	
_AHCI_SDBFIS	47	Set Device Bits FIS	
_AHCI_UFIS	48	Unknown FIS (up to 64 bytes)	
_AHCI_RF_RESERVED	49	Reserved	
_AHCI_CFIS	50	Command FIS	_AHCI_CMND_TABLE
_AHCI_ACMD	51	ATAPI Command	
_AHCI_CT_RESERVED	52	Reserved	
_AHCI_PRDT	53	Physical Region Descriptor Table	
_AHCI_DATA	58	Actual data pointed by DBA && DBAU	

in.AHCIRegIdAsString: Contains an AHCI register name.

in.PortNum: Contains a port number.

in.SlotNum: Contains a slot number.

in.AHCITraHasError: If set to a non-zero value, indicates AHCI transaction has errors.

in.AHCIErrorId: Contains the numeric encoding of the AHCI error type. The following possible values are defined by VSE and the corresponding constants can be used by scripts:

Error type	Value	Error name	Error description
_AHCI_ERROR_RESERVED_NOT_NULL	0	Reserved field is not 0	Reserved registers and fields must be 0 for AHCI
_AHCI_ERROR_INCOMPLETE_TRA	1	Incomplete	Incomplete AHCI transaction
_AHCI_ERROR_LOGICAL_ERROR	2	Logical error	Discrepancy between the states of different registers, wrong order of transactions, etc.
_AHCI_ERROR_RO_VIOLATION	3	Read-only field changed	An attempt of writing to the read-only fields and registers
_AHCI_ERROR_COMPLETER_ABORT	4	Completer abort	Completer abort transaction
_AHCI_ERROR_TRA_STARTS_FROM_MIDDLE	5	Unexpected register offset	Incomplete AHCI transaction, that starts from the middle of register
_AHCI_ERROR_REG_ACCESS_VIOLATION	6	Register access violation	HBA register access is longer than 64 bits or crosses 8-byte alignment boundary
_AHCI_ERROR_TABLE_ALIGNMENT_ERROR	7	Address is not properly aligned	Table base address is not properly aligned in memory
_AHCI_ERROR_INVALID_HBA_STATE	8	Invalid HBA state	An attempt to do something forbidden in current HBA state.
_AHCI_ERROR_INVALID_FIS_SIZE	9	Invalid FIS size	FIS size doesn't correspond to spec or CFIS.CFL field.
_AHCI_ERROR_UNKNOWN	10	Unknown error	All other errors.
_AHCI_ERROR_NO_ERROR	12	No errors	Correct transaction

in.AHCIErrorIdAsString: Contains an AHCI error name

The following table shows the list of AHCI registers and their fields defined in the input context. The fields can be accessed by using "_". E.g. **in.CAP_SXS** contains the numeric encoding of the SXS field of CAP register.

Note: If length of returned value is bigger than 1 dword, please, specify dword by using “_DW” and dword index, otherwise string in a hex format will be returned. For example **in.HBA_VENDOR_SPECIFIC_DW3** contains the numeric encoding of the 3rd dword of HBA Vendor Specific field.

Parameter	Fields	Offset	Meaning
CAP	NP	04:00	Number of Ports
	SXS	05	Supports External SATA
	EMS	06	Enclosure Management Supported
	CCCS	07	Command Completion Coalescing Supported
	NCS	12:08	Number of Command Slots
	PSC	13	Partial State Capable
	SSC	14	Slumber State Capable
	PMD	15	PIO Multiple DRQ Block (PMD)
	FBSS	16	FIS-based Switching Supported
	SPM	17	Supports Port Multiplier
	SAM	18	Supports AHCI mode only
	RSVD	19	Reserved
	ISS	23:20	Interface Speed Support
	SCLO	24	Supports Command List Override
	SAL	25	Supports Activity LED
	SALP	26	Supports Aggressive Link Power Management
	SSS	27	Supports Staggered Spin-up
	SMPS	28	Supports Mechanical Presence Switch
	SSNTF	29	Supports SNotification Register
	SNCQ	30	Supports Native Command Queuing
	S64A	31	Supports 64-bit Addressing
GHC	HR	00	HBA Reset
	IE	01	Interrupt Enable
	MRSM	02	MSI Revert to Single Message
	RSVD	30:03	Reserved
	AE	31	AHCI Enable
IS	IPS	31:00	Interrupt Pending Status
PI	PI	31:00	Port Implemented
VS	MNR	15:00	Minor Version Number
	MJR	31:16	Major Version Number
CCC_CTL	EN	0	Enable
	RSVD	2:1	Reserved
	INT	7:3	Interrupt
	CC	15:8	Command Completions
	TV	31:16	Timeout Value
CCC_PORTS	PRT	31:0	Ports
EM_LOC	SZ	15:0	Buffer Size

	OFST	31:16	Offset
EM_CTL	STS_MR	0	Message Received
	RSVD4	07:01	Reserved
	CTL_TM	8	Transmit Message
	CTL_RST	9	Reset
	RSVD3	15:10	Reserved
	SUPP_LED	16	LED Message Types
	SUPP_SAFTE	17	SAF-TE Enclosure Management Messages
	SUPP_SES2	18	SES-2 Enclosure Management Messages
	SUPP_SGPIO	19	SGPIO Enclosure Management Messages
	RSVD2	23:20	Reserved
	ATTR_SMB	24	Single Message Buffer
	ATTR_XMT	25	Transmit Only
	ATTR_ALHD	26	Activity LED Hardware Driven
	ATTR_PM	27	Port Multiplier Support
	RSVD	31:28	Reserved
CAP2	BOH	00	BIOS/OS Handoff
	NVMP	01	NVMHCI Present
	APST	02	Automatic Partial to Slumber Transitions
	SDS	03	Supports Device Sleep
	SADM	04	Supports Aggressive Device Sleep Management
	DESO	05	DevSleep Entrance from Slumber Only
	RSVD	31:06	Reserved
BOHC	BOS	00	BIOS Owned Semaphore
	OOS	01	OS Owned Semaphore
	SOOE	02	SMI on OS Ownership Change Enable
	OOC	03	OS Ownership Change
	BB	04	BIOS Busy
	RSVD	31:05	Reserved
RSVD_HBA	-		Reserved
RSVD_NVMHCI	-		Reserved for NVMHCI
HBA_VENDOR_SPECIFIC	-		Venfor specific
Port registers			
PxCLB	RSVD	09:00	Reserved
	CLB	31:10	Command List Base Address
PxCLBU	CLBU	31:00	Command List Base Address Upper
PxFB	RSVD	07:00	Reserved
	FB	31:08	FIS Base Address

PxFBU	FBU	31:00	FIS Base Address Upper
PxIS	DHRS	00	Device to Host Register FIS Interrupt
	PSS	01	PIO Setup FIS Interrupt
	DSS	02	DMA Setup FIS Interrupt
	SDBS	03	Set Device Bits Interrupt
	UFS	04	Unknown FIS Interrupt
	DPS	05	Descriptor Processed
	PCS	06	Port Connect Change Status
	DMPS	07	Device Mechanical Presence Status
	RSVD2	21:08	Reserved
	PRCS	22	PhyRdy Change Status
	IPMS	23	Incorrect Port Multiplier Status
	OFS	24	Overflow Status
	RSVD1	25	Reserved
	INFS	26	Interface Non-fatal Error Status
	IFS	27	Interface Fatal Error Status
	HBDS	28	Host Bus Data Error Status
	HBFS	29	Host Bus Fatal Error Status
	TFES	30	Task File Error Status
	CPDS	31	Cold Port Detect Status
PxIE	DHRE	00	Device to Host Register FIS Interrupt Enable
	PSE	01	PIO Setup FIS Interrupt Enable
	DSE	02	DMA Setup FIS Interrupt Enable
	SDBE	03	Set Device Bits FIS Interrupt Enable
	UFE	04	Unknown FIS Interrupt Enable
	DPE	05	Descriptor Processed Interrupt Enable
	PCE	06	Port Change Interrupt Enable
	DMPE	07	Device Mechanical Presence Enable
	RSVD2	21:08	Reserved
	PRCE	22	PhyRdy Change Interrupt Enable
	IPME	23	Incorrect Port Multiplier Enable
	OFE	24	Overflow Enable
	RSVD	25	Reserved
	INFE	26	Interface Non-fatal Error Enable
	IFE	27	Interface Fatal Error Enable
	HBDE	28	Host Bus Data Error Enable
	HBFE	29	Host Bus Fatal Error Enable
	TFEE	30	Task File Error Enable

	CPDE	31	Cold Presence Detect Enable
PxCMD	ST	00	Start
	SUD	01	Spin-Up Device
	POD	02	Power On Device
	CLO	03	Command List Override
	FRE	04	FIS Receive Enable
	RSVD	07:05	Reserved
	CSS	12:08	Current Command Slot
	MPSS	13	Mechanical Presence Switch State
	FR	14	FIS Receive Running
	CR	15	Command List Running
	CPS	16	Cold Presence State
	PMA	17	Port Multiplier Attached
	HPCP	18	Hot Plug Capable Port
	MPSP	19	Mechanical Presence Switch Attached to Port
	CPD	20	Cold Presence Detection
	ESP	21	External SATA Port
	FBSCP	22	FIS-based Switching Capable Port
	APSTE	23	Automatic Partial to Slumber Transitions Enabled
	ATAPI	24	Device is ATAPI
	DLAE	25	Drive LED on ATAPI Enable
	ALPE	26	Aggressive Link Power Management Enable
	ASP	27	Aggressive Slumber / Partial
	ICC	31:28	Interface Communication Control
PxRSVD	-		Reserved
PxTFD	STS_ERR	00	Error during the transfer.
	STS_CS1	02:01	Command specific
	STS_DRQ	03	Data transfer is requested
	STS_CS2	06:04	Command specific
	STS_BSY	07	Interface is busy
	ERR	15:08	Error
	RSVD	31:16	Reserved
PxSIG	SC	07:00	Sector Count Register
	LBA_LOW	15:08	LBA Low Register
	LBA_MID	23:16	LBA Mid Register
	LBA_HIGH	31:24	LBA High Register
PxSSTS	DET	03:00	Device Detection
	SPD	07:04	Current Interface Speed

	IPM	11:08	Interface Power Management
	RSVD	31:12	Reserved
PxSCTL	DET	03:00	Device Detection Initialization
	SPD	07:04	Speed Allowed
	IPM	11:08	Interface Power Management Transitions Allowed
	SPM	15:12	Select Power Management
	PMP	19:16	Port Multiplier Port
	RSVD	31:20	Reserved
PxSERR	ERR_I	00	Recovered Data Integrity Error
	ERR_M	01	Recovered Communications Error
	ERR_RSVD2	07:02	Reserved
	ERR_T	08	Transient Data Integrity Error
	ERR_C	09	Persistent Communication or Data Integrity Error
	ERR_P	10	Protocol Error
	ERR_E	11	Internal Error
	ERR_RSVD	15:12	Reserved
	DIAG_N	16	PhyRdy Change
	DIAG_I	17	Phy Internal Error
	DIAG_W	18	Comm Wake
	DIAG_B	19	10B to 8B Decode Error
	DIAG_D	20	Disparity Error
	DIAG_C	21	CRC Error
	DIAG_H	22	Handshake Error
	DIAG_S	23	Link Sequence Error
	DIAG_T	24	Transport state transition error
	DIAG_F	25	Unknown FIS Type
	DIAG_X	26	Exchanged
	DIAG_RSVD	31:27	Reserved
PxSACT	DS	31:00	Device Status
PxCI	CI	31:00	Commands Issued
PxSNTF	PMN	15:00	PM Notify
	RSVD	31:16	Reserved
PxFBS	EN	00	Enable
	DEC	01	Device Error Clear
	SDE	02	Single Device Error
	RSVD2	07:03	Reserved
	DEV	11:08	Device To Issue
	ADO	15:12	Active Device Optimization

	DWE	19:16	Device With Error
	RSVD	31:20	Reserved
PxDEVSLP	ADSE	00	Aggressive Device Sleep Enable
	DSP	01	Device Sleep Present
	DETO	09:02	Device Sleep Exit Timeout
	MDAT	14:10	Minimum Device Sleep Assertion Time
	DITO	24:15	Device Sleep Idle Timeout
	DM	28:25	DITO Multiplier
	RSVD	31:29	Reserved
PxRSVD2	-		Reserved
PxVS	-		Vendor Specific
FIS registers			
RFIS	TYPE	07:00	FIS Type
	PMP	11:08	The Port Multiplier Port
	RSVD4	13:12	Reserved
	I	14	Interrupt bit
	RSVD3	15	Reserved
	STATUS	23:16	Status
	ERROR	31:24	Error - Contains the new value of the Er
	LBA_LO	39:32	LBA(7:0)
	LBA_MID	47:40	LBA(15:8)
	LBA_HI	55:48	LBA(23:16)
	DEVICE	63:56	Device
	LBA_LO_EXP	71:64	LBA(31:24)
	LBA_MID_EXP	79:27	LBA(39:32)
	LBA_HI_EXP	87:80	LBA(47:40)
	RSVD2	95:88	Reserved
	COUNT	103:96	Count(7:0)
	COUNT_EXP	111:104	Count(15:8)
	RSVD	127:112	Reserved
DSFIS	TYPE	07:00	FIS Type
	PMP	11:08	The Port Multiplier Port
	RSVD4	12	Reserved
	D	13	Direction
	I	14	Interrupt
	A	15	Auto-Activate
	RSVD3	31:16	Reserved

	DMA_BI_LO	63:32	DMA Buffer Identifier Low
	DMA_BI_HI	95:64	DMA Buffer Identifier High
	RSVD2	127:96	Reserved
	BO	159:128	DMA Buffer Offset
	TC	191:160	DMA Transfer Count
	RSVD	223:192	Reserved
PSFIS	TYPE	07:00	FIS Type
	PMP	11:08	The Port Multiplier Port
	RSVD5	12	Reserved
	D	13	Direction
	I	14	Interrupt
	RSVD4	15	Reserved
	STATUS	23:16	Status
	ERROR	31:24	Error
	LBA_LO	39:32	LBA(7:0)
	LBA_MID	47:40	LBA(15:8)
	LBA_HI	55:48	LBA(23:16)
	DEVICE	63:56	Device
	LBA_MID_EXP	71:64	LBA(39:32)
	LBA_HI_EXP	79:72	LBA(47:40)
	RSVD3	87:80	Reserved
	COUNT	95:88	Count(7:0)
	COUNT_EXP	103:96	Count(15:8)
	RSVD2	111:104	Reserved
	E_STATUS	119:112	E_Status
	TC	135:120	Transfer Count
	RSVD	151:136	Reserved
SDBFIS	TYPE	07:00	FIS Type
	PMP	11:08	The Port Multiplier Port
	RSVD3	13:12	Reserved
	I	14	Interrupt Bit
	N	15	Notification Bit
	STATUS_LO	18:16	Status-Lo
	RSVD2	19	Reserved
	STATUS_HI	22:20	Status-Hi
	RSVD	23	Reserved
	ERROR	31:24	Error
	PROT_SPEC	64:32	Protocol Specific

Command Table			
PRDT	DBA_RSVD	00	Reserved
	DBA	31:01	Data Base Address
	DBAU	63:32	Data Base Address Upper 32-bits
	RSVD	95:64	Reserved
	DI_DBC	117:96	Data Byte Count
	DI_RSVD	126:118	Reserved
	DI_I	127	Description Information: Interrupt on Completion
ACMD	ACMD	-	Description Information: ATAPI Command
Command List			
COMMAND_HEADER	CFL	04:00	Command FIS Length
	A	05	ATAPI
	W	06	Write
	P	07	Prefetchable
	R	08	Reset
	B	09	BIST
	C	10	Clear Busy upon R_OK
	RSVD	11	Reserved
	PMP	15:12	Port Multiplier Port
	PRDTL	31:16	Physical Region Descriptor Table Length
	PRDBC	63:32	Physical Region Descriptor Byte Count
	CTBA_RSVD	70:64	Reserved
	CTBA	95:71	Command Table Descriptor Base Address
	CTBAU	127:96	Command Table Descriptor Base Address Upper 32-bits
	RSVD1	159:128	Reserved
	RSVD2	191:160	Reserved
	RSVD3	223:192	Reserved
	RSVD4	255:224	Reserved

5.3.1 Metric values

in.Metric_Throughput: Metric presenting transaction payload divided by response time, expressed in **kilobytes** per second, an integer value

in.Metric_PayloadBytes: Metric presenting number of data payload bytes this NVM Transaction transferred, an integer value.

in.Metric_NumOfLinkAndSplitTras: Metric presenting the total number of Link and Split Transactions that compose this AHCI Transaction, an integer value.

in. Metric_ResponseTime: Metric presenting time it took to transmit this AHCI Transaction on the link, from the beginning of the first packet to the end of the last packet in the transaction, a VSE time object value (see 9.1 VSE Time Object for details)

5.4 ATA transaction-specific set of members

Valid for ATA transactions only. Undefined for other events.

All the ATA-specific values are present in the input context for ATA transactions, depending upon the type of register. Also the common PayloadLength and Payload values reflect the total combined payload for the ATA transaction. In addition to that, the following values exist:

in.ataCommandCode: Contains ATA command code value.

in.ataCommand: Contains the numeric encoding of the ATA command id. The following possible values are defined by VSE and the corresponding constants can be used by scripts:

Constant	Value
_ATA_NOP	0
_ATA_RESERVED	1
_ATA_CFA_REQUEST_EXTENDED_ERROR	2
_ATA_DATA_SET_MANAGEMENT	3
_ATA_DEVICE_RESET	4
_ATA_REQUEST_SENSE_DATA_EXT	5
_ATA_OBSOLETE	6
_ATA_RETIRED	7
_ATA_READ_SECTORS	8
_ATA_READ_SECTORS_EXT	9
_ATA_READ_DMA_EXT	10
_ATA_READ_NATIVE_MAX_ADDRESS_EXT	11
_ATA_READ_MULTIPLE_EXT	12
_ATA_READ_STREAM_DMA_EXT	13
_ATA_READ_STREAM_EXT	14
_ATA_READ_LOG_EXT	15
_ATA_WRITE_SECTORS	16
_ATA_WRITE_SECTORS_EXT	17
_ATA_WRITE_DMA_EXT	18
_ATA_SET_MAX_ADDRESS_EXT	19
_ATA_CFA_WRITE_SECTORS_WITHOUT_ERASE	20
_ATA_WRITE_SECTORS	21
_ATA_WRITE_SECTORS_EXT	22
_ATA_WRITE_STREAM_EXT	23
_ATA_WRITE_DMA_FUA_EXT	24
_ATA_WRITE_LOG_EXT	25
_ATA_READ_VERIFY_SECTORS	26

_ATA_READ_VERIFY_SECTORS_EXT	27
_ATA_WRITE_UNCORRECTABLE_EXT	28
_ATA_READ_LOG_DMA_EXT	29
_ATA_CONFIGURE_STREAM	30
_ATA_WRITE_LOG_DMA_EXT	31
_ATA_TRUSTED_NON_DATA	32
_ATA_TRUSTED_RECEIVE	33
_ATA_TRUSTED_RECEIVE_DMA	34
_ATA_TRUSTED_SEND	35
_ATA_TRUSTED_SEND_DMA	36
_ATA_READ_FPDMA_QUEUED	37
_ATA_WRITE_FPDMA_QUEUED	38
_ATA_VENDOR_SPECIFIC	39
_ATA_CFA_TRANSLATE_SECTOR	40
_ATA_EXECUTE_DEVICE_DIAGNOSTIC	41
_ATA_DOWNLOAD_MICROCODE	42
_ATA_DOWNLOAD_MICROCODE_DMA	43
_ATA_PACKET	44
_ATA_IDENTIFY_PACKET_DEVICE	45
_ATA_SMART	46
_ATA_DEVICE_CONFIGURATION_OVERLAY	47
_ATA_SANITIZE_DEVICE	48
_ATA_NV_CACHE	49
_ATA_RESERVED_FOR_THE_COMPACTFLASH_ASSOCIATION	50
_ATA_CFA_ERASE_SECTORS	51
_ATA_READ_MULTIPLE	52
_ATA_WRITE_MULTIPLE	53
_ATA_SET_MULTIPLE_MODE	54
_ATA_READ_DMA	55
_ATA_WRITE_DMA	56
_ATA_CFA_WRITE_MULTIPLE_WITHOUT_ERASE	57
_ATA_WRITE_MULTIPLE_FUA_EXT	58
_ATA_CHECK_MEDIA_CARD_TYPE	59
_ATA_RESERVED_FOR_THE_MEDIA_CARD_PASS_THROUGH_COMMAND_FEATURE_SET	60
_ATA_STANDBY_IMMEDIATE	61
_ATA_IDLE_IMMEDIATE	62
_ATA_STANDBY	63

_ATA_IDLE	64
_ATA_READ_BUFFER	65
_ATA_CHECK_POWER_MODE	66
_ATA_SLEEP	67
_ATA_FLUSH_CACHE	68
_ATA_WRITE_BUFFER	69
_ATA_READ_BUFFER_DMA	70
_ATA_FLUSH_CACHE_EXT	71
_ATA_WRITE_BUFFER_DMA	72
_ATA_IDENTIFY_DEVICE	73
_ATA_SET_FEATURES	74
_ATA_SECURITY_SET_PASSWORD	75
_ATA_SECURITY_UNLOCK	76
_ATA_SECURITY_ERASE_PREPARE	77
_ATA_SECURITY_ERASE_UNIT	78
_ATA_SECURITY_FREEZE_LOCK	79
_ATA_SECURITY_DISABLE_PASSWORD	80
_ATA_READ_NATIVE_MAX_ADDRESS	81
_ATA_SET_MAX_ADDRESS	82

in.ataProtocol: Contains the numeric encoding of the ATA protocol id. The following possible values are defined by VSE and the corresponding constants can be used by scripts:

Constant	Value	Description
_ATA_PROTOCOL_NONE	0	Protocol undefined
_ATA_PROTOCOL_ND	0x1	Non-Data command
_ATA_PROTOCOL_PI	0x2	PIO Data-In command
_ATA_PROTOCOL_PO	0x4	PIO Data-Out command
_ATA_PROTOCOL_DM	0x8	DMA command
_ATA_PROTOCOL_DMQ	0x10	DMA QUEUED command
_ATA_PROTOCOL_DR	0x20	DEVICE RESET command
_ATA_PROTOCOL_DD	0x40	EXECUTE DEVICE DIAGNOSTIC command
_ATA_PROTOCOL_P	0x80	PACKET command
_ATA_PROTOCOL_VS	0x100	Vendor specific

in.ataTraHasError: If set to a non-zero value, indicates ATA command has errors.

in.ataErrorId: Contains the numeric encoding of the ATA error type. The following possible values are defined by VSE and the corresponding constants can be used by scripts:

Error type	Value	Error name	Error description

_ATA_ERROR_WRONG_SUB_TRA	0	Error in Sub Transaction	ATA command contains sub-transaction with error in it (for example AHCI)
_ATA_ERROR_INCOMPLETE_SUB_TRA	1	Incomplete Sub Transaction	ATA command contains incomplete sub-transaction (for example AHCI)
_ATA_ERROR_INCOMPLETE_TRA	2	Incomplete transaction	Incomplete ATA transaction
_ATA_ERROR_LOGICAL_ERROR	3	Logical error	Wrong command order, wrong amount of data transferred, etc.
_ATA_ERROR_ERROR_BIT_SET	4	Error bit set	Command has error bit set to 1
_ATA_ERROR_SHADOW_ERROR	5	Shadow error bit set	Shadow error byte in AHCI is set
_ATA_ERROR_NO_ERROR	7	No error	Correct transaction

in.ATAErrorIdAsString: Contains an ATA error name

in.ataPort: Contains port number.

in.ataSlot: Contains slot number.

in.ataFeatures: Contains feature sector count.

in.ataCount: Contains the numeric encoding of the Sector Count register of the Shadow Register Block.

in.ataLBA: Contains the numeric encoding of the LBA low, mid and high registers of the Shadow Register Block

in.ataLBAEXT: Contains the numeric encoding of the expanded LBA register of the Shadow Register Block.

in.ataDevice: Contains the first byte of the Device register of the Shadow Register Block.

in.ataDevice0: Returns first bit of in.ataDevice value.

in.ataDevice7: Returns last bit of in.ataDevice value.

in.ataOutputStatus: Contains the numeric encoding of transaction status register.

in.ataBSY: If set to a non-zero value, indicates that the device is accessing the registers Indicates the interface is busy.

in.ataDRDY: If set to a non-zero value, indicatesthat the device is capable of responding to a command.

in.ataDF: If set to a non-zero value, indicates that the device has detected a write fault condition.

in.ataDSC: If set to a non-zero value, indicates that a seek has been completed and the device head is settled over a track.

in.ataDRQ:Data If set to a non-zero value, indicates that the device is ready to transfer a word or byte of data between the host and the device.

in.ataCORR: If set to a non-zero value, indicates that a correctable data error was encountered and the data has been corrected. This condition does not terminate a data transfer.

in.ataIDX: Is set to a non-zero value once per revolution.

in.ataERR: If set to a non-zero value, indicates an error during the transfer

in.ataPRIO: If set to a non-zero value, then the command shall be high priority. Otherwise, the command shall be normal priority.

in.ataNCQ: Contains the numeric encoding of the NCQ Tag field.

in.ataRARC: Returns the RARC bit

in.ataICC: Contains the numeric encoding of the Isochronous Command Completion field.

in.ataHybridInfo: If set to a non-zero value, then the device supports the hybrid information feature.

in.ataPayloadLength: Contains the exact payload size (transferred data) in bytes.

in.ataRequestedBytesCount: Contains the requested data size in bytes

in.ataHasData: If set to a non-zero value, indicates payload presence.

in.ataIsDeviceToHostTransition: If set to a non-zero value, indicates the Host to Device direction.

in.ataHasErrors: If set to a non-zero value, indicates an error in decoded transaction.

in.ataContainsPxSACT: If set to a non-zero value, indicates PxSACT register usage

in.ataContainsCFIS: If set to a non-zero value, indicates CFIS register usage

in.ataContainsSDBFIS: If set to a non-zero value, indicates SDBFIS register usage

in.ataContainsPSFIS: If set to a non-zero value, indicates PSFIS register usage

in.ataContainsRFIS: If set to a non-zero value, indicates RFIS register usage

in.ataContainsInterruptD2H: If set to a non-zero value, indicates Device-To-Host interrupt happened

in.ataContainsInterruptH2D: If set to a non-zero value, indicates Host-To-Device interrupt happened

in.ataContainsIncompleteSubTra: If set to a non-zero value, indicates ATA transaction has incomplete sub-transactions

in.ataContainsErrorSubTra: If set to a non-zero value, indicates ATA transaction has sub-transactions with errors

in.ataHasShadowRegister: If set to a non-zero value, indicates a RFIS Shadow Register presence

in.ataRFISShadowRegister: Contains the numeric encoding of the RFIS Shadow Register

5.4.1 Metric values

in.Metric_Throughput: Metric presenting transaction payload divided by response time, expressed in **kilobytes** per second, an integer value

in.Metric_PayloadBytes: Metric presenting number of data payload bytes this NVM Transaction transferred, an integer value.

in.Metric_NumOfAHCITras: Metric presenting the total number of AHCI Transactions that compose this ATA Transaction, an integer value.

in. Metric_ResponseTime: Metric presenting time it took to transmit this ATA Transaction on the link, from the beginning of the first packet to the end of the last packet in the transaction, a VSE time object value (see 9.1 VSE Time Object for details)

5.5 MCTP Messages transaction-specific set of members

Valid for MCTP Messages transactions only. Undefined for other messages.

5.5.1 Any message types

All the MCTP Messages specific values are present in the input context for MCTP transactions. The following values exist for any message types:

in.DstEndpointId: The EID for the endpoint to receive the MCTP packet. A few EID values are reserved for specific routing.

in.IC: (MCTP integrity check bit) Indicates whether the MCTP message is covered by an overall MCTP message payload integrity check. This field is required to be the most significant bit of the first byte of the message body in the first packet of a message along with the message type bits.

```
0b = No MCTP message integrity check
1b = MCTP message integrity check is present
```

in.IsComplete: Ehe message starts with packet with SOM bit set, ends with packet with EOM bit set, and all Ptk Seq # numbers for its packets are sequential. This means that all the packets of this message are present.

in.IsNoEnd: This message does not contain packet with EOM bit set.

in.IsNoSeq: Ptk Seq # field for a packet does not equal to Ptk Seq # for its previous packet plus 1 mod 4.

in.IsNoStart: This message does not contain packet with SOM bit set.

in.SrcEndpointId: The EID of the originator of the MCTP packet.

in.TO: The Tag Owner (TO) bit identifies whether the message tag was originated by the endpoint that is the source of the message or by the endpoint that is the destination of the message. The Message Tag field is generated and tracked independently for each value of the Tag Owner bit. MCTP message types may overlay this bit with additional meaning, for example using it to differentiate between "request" messages and "response" messages.

in.Type: Defines the type of payload contained in the message data portion of the MCTP message. This field is required to be contained in the least-significant bits of the first byte of the message body in the first packet of a message. Like the fields in the MCTP transport header, the message type field is one of the common MCTP fields that are present independent of the transport over which MCTP is being used. Unlike the MCTP transport header, however, the message type field is only required to be present in the first packet of a particular MCTP message, whereas the MCTP transport header fields are present in every MCTP packet. The following possible values are defined by VSE and the corresponding constants can be used by scripts:

Constant	Code	Type
_MCTP_MSG_CONTROL	0x00	MCTP Control
_MCTP_MSG_PLDM	0x01	Platform Level Data Model (PLDM)
_MCTP_MSG_NC_SI_OVER_MCTP	0x02	NC-SI over MCTP
_MCTP_MSG_ETHERNET_OVER_MCTP	0x03	Ethernet over MCTP
_MCTP_MSG_NVME_OVER_MCTP	0x04	NVM Express Management Messages over MCTP
_MCTP_MSG_VENDOR_DEFINED_PCI	0x7E	Vendor Defined – PCI

_MCTP_MSG_VENDOR_DEFINED_IANA	0x7F	Vendor Defined – IANA
_MCTP_MSG_RESERVED	0xFF	Reserved

in.Errors: The List of errors present in this transaction. If there are no errors, it's an empty list.

Constant	Code	Type
_MCTP_MSG_ERROR_RESERVED_NOT_NULL	0x02	Reserved field is not null
_MCTP_MSG_ERROR_INVALID_FIELD_VALUE	0x04	Field has an invalid value
_MCTP_MSG_ERROR_INVALID_LENGTH	0x08	Transaction has invalid length
_MCTP_MSG_ERROR_INCOMPLETE_TRA	0x10	Transaction is incomplete

Note: in.Pad: Pad Length value for MCTP VDM packets, zero value will be returned for SMBus packets.

5.5.2 Control Messages

The following values are valid for Control Messages only. Undefined for other messages.

in.CommandCode: For Request messages, this field is a command code indicating the type of MCTP operation the packet is requesting. The Command Code that is sent in a Request must be returned in the corresponding Response. The following possible values are defined by VSE:

Constant	Code	Type
_MCTP_COMMAND_CODE_Reserved	0x00	Reserved
_MCTP_COMMAND_CODE_Set_Endpoint_ID	0x01	Set Endpoint ID
_MCTP_COMMAND_CODE_Get_Endpoint_ID	0x02	Get Endpoint ID
_MCTP_COMMAND_CODE_Get_Endpoint_UUID	0x03	Get Endpoint UUID
_MCTP_COMMAND_CODE_Get_MCTP_Version_Support	0x04	Get MCTP Version Support
_MCTP_COMMAND_CODE_Get_Message_Type_Support	0x05	Get Message Type Support
_MCTP_COMMAND_CODE_Get_Vendor_Defined_Message_Support	0x06	Get Vendor Defined Message Support
_MCTP_COMMAND_CODE_Resolve_Endpoint_ID	0x07	Resolve Endpoint ID
_MCTP_COMMAND_CODE_Allocate_Endpoint_IDs	0x08	Allocate Endpoint IDs
_MCTP_COMMAND_CODE_Routing_Information_Update	0x09	Routing Information Update
_MCTP_COMMAND_CODE_Get_Routing_Table_Entries	0x0A	Get Routing Table Entries
_MCTP_COMMAND_CODE_Prepare_for_Endpoint_Discovery	0x0B	Prepare for Endpoint Discovery
_MCTP_COMMAND_CODE_Endpoint_Discovery	0x0C	Endpoint Discovery
_MCTP_COMMAND_CODE_Discovery_Notify	0x0D	Discovery Notify
_MCTP_COMMAND_CODE_Get_Network_ID	0x0E	Get Network ID
_MCTP_COMMAND_CODE_Query_Hop	0x0F	Query Hop
_MCTP_COMMAND_CODE_Resolve_UUID	0x10	Resolve UUID

in.CompletionCode: The command/result code field is used to return management operation results for response messages:

Constant	Code	Type
_MCTP_MSG_RESP_CODE_SUCCESS	0x00	SUCCESS
_MCTP_MSG_RESP_CODE_ERROR	0x01	ERROR
_MCTP_MSG_RESP_CODE_ERROR_INVALID_DATA	0x02	ERROR_INVALID_DATA
_MCTP_MSG_RESP_CODE_ERROR_INVALID_LENGTH	0x03	ERROR_INVALID_LENGTH
_MCTP_MSG_RESP_CODE_ERROR_NOT_READY	0x04	ERROR_NOT_READY
_MCTP_MSG_RESP_CODE_ERROR_UNSUPPORTED_CMD	0x05	ERROR_UNSUPPORTED_CMD
	0x80 … 0xFF	COMMAND_SPECIFIC
	all	Reserved

	other	

in.ControlType: The different types of messages shown below:

Constant	Code	Type
_MCTP_MSG_CTRL_TYPE_BROADCAST_DATAGRAM	0x00	Broadcast Datagram
_MCTP_MSG_CTRL_TYPE_BROADCAST_REQUEST	0x01	Broadcast Request
_MCTP_MSG_CTRL_TYPE_DATAGRAM	0x02	Datagram
_MCTP_MSG_CTRL_TYPE_REQUEST	0x03	Request
_MCTP_MSG_CTRL_TYPE_RESPONSE	0x04	Response
_MCTP_MSG_CTRL_TYPE_UNKNOWN	0x05	Unknown

in.DBit: Datagram bit. This bit is used to indicate whether the Instance ID field is being used for tracking and matching requests and responses, or is just being used to identify a retransmitted message.

in.InstanceId: The Instance ID field is used to identify new instances of an MCTP control Request or Datagram to differentiate new requests or datagrams that are sent to a given message terminus from retried messages that are sent to the same message terminus. The Instance ID field is also used to match up a particular instance of an MCTP Response message with the corresponding instance of an MCTP Request message.

in.RqBit: Request bit. This bit is used to help differentiate between MCTP control Request messages and other message classes.

5.5.2.1 Set Endpoint ID Message

The following values are valid for Set Endpoint ID messages only. Undefined for other messages. See Specification for field detailed descriptions.

Request fields:

in.EID: Endpoint ID.

in.Operation: Operation.

Response fields:

in.EID_Assignment_Status: EID assignment status.

in.EID_Allocation_Status: Endpoint ID allocation status.

in.EID_Pool_Size: EID Pool Size.

5.5.2.2 Get Endpoint ID Message

The following values are valid for Get Endpoint ID messages only. Undefined for other messages. See Specification for field detailed descriptions.

Response fields:

in.EID: Endpoint ID.

in.Endpoint_Type: Endpoint Type.

in.EID_Type: Endpoint ID Type.

in.Medium_Specific: Medium-Specific Information.

5.5.2.3 Get Endpoint UUID

The following values are valid for Get Endpoint UUID messages only. Undefined for other messages. See Specification for field detailed descriptions.

Response fields:

in.UUID: UUID string.

5.5.2.4 Get MCTP Version Support

The following values are valid for Get MCTP Version Support messages only. Undefined for other messages. See Specification for field detailed descriptions.

Request fields:

in.Message_Type_Ex: The Message Type Number to retrieve version information.

Response fields:

in.Entries_Count: One-based Version Number Entry count. If in.Entries_Count is equal to 5, 6 version numbers are returned.

in.Major_Version: Array with major version numbers.

in.Minor_Version: Array with minor version numbers.

in.Update_Version: Array with update version numbers.

in.Alpha_Version: Array with “alpha” bytes.

5.5.2.5 Get Message Type Support

The following values are valid for Get Message Type Support messages only. Undefined for other messages. See Specification for field detailed descriptions.

Response fields:

in.Entries_Count: One-based MCTP Message Type count. If in.Entries_Count is equal to 5, 6 message types are returned.

in.Message_Type: Array with message type numbers.

5.5.2.6 Get Vendor Defined Message Support

The following values are valid for Get Vendor Defined Message only. Undefined for other messages. See Specification for field detailed descriptions.

Request fields:

in.VID_Set_Selector: Vendor ID Set Selector.

Response fields:

in.VID_Set_Selector: Vendor ID Set Selector.

in.VID_Format: Vendor ID format.

in.VID_PCIe: PCI Vendor ID.

in.VID_IANA: IANA Enterprise Number.

in.VID_Specific: Vendor specific data.

5.5.2.7 Resolve Endpoint ID Support

The following values are valid for Resolve Endpoint ID messages only. Undefined for other messages. See Specification for field detailed descriptions.

Request fields:

in.EID: Endpoint ID.

Response fields:

in.EID: Endpoint ID.

in.BDF: Bus Device Function.

5.5.2.8 Allocate Endpoint IDs Support

The following values are valid for Allocate Endpoint IDs messages only. Undefined for other messages. See Specification for field detailed descriptions.

Request fields:

in.Operation_Flags: Operation Flags.

in.Endpoints_Count: Number of Endpoint IDs (Allocated Pool Size).

in.EID: Endpoint ID.

Response fields:

in.Pool_Allocation_Status: Pool allocation status.

in.EID: Endpoint ID.

5.5.2.9 Rebuilding Routing Tables Support

The following values are valid for Rebuilding Routing Tables messages only. Undefined for other messages. See Specification for field detailed descriptions.

Request fields:

in.Entries_Count: One-based MCTP Message Type count. If in.Entries_Count is equal to 5, 6 message types are returned.

in.Entry_Type: Array with Entry Types.

in.Endpoints_Count_Table: Array of sizes of EID Range.

in.EID: Array of endpoint ID.

in.BDF: array of bus Device Function.

5.5.2.10 Get Routing Table Entries Support

The following values are valid for Get Routing Table Entries messages only. Undefined for other messages. See Specification for field detailed descriptions.

Request fields:

in.Entry_Handle: Entry Handle.

Response fields:

in.Next_Entry_Handle: Next Entry Handle.

in.Entries_Count: One-based MCTP Message Type count. If in.Entries_Count is equal to 5, 6 message types are returned.

in.Endpoints_Count_Table: Array of sizes of EID Range.

in.Entry_Type: Array of entry types.

in.Dynamic_Static: Array of dynamic/static entries.

in.Port: Array of port numbers.

in.Transport_Binding: Array of Physical Transport Binding Identifiers.

in.Media_Type_ID: Array of Physical Media Type Identifiers.

in.Address_Size: Array of Physical Address Sizes.

in.Address: Array of Physical Addresses.

5.5.2.11 Get Network ID Support

The following values are valid for Get Network ID messages only. Undefined for other messages. See Specification for field detailed descriptions.

Response fields:

in.UUID: Network ID bytes.

5.5.2.12 Query Hop Support

The following values are valid for Query Hop messages only. Undefined for other messages. See Specification for field detailed descriptions.

Request fields:

in.EID: Endpoint ID.

in.Message_Type: Message type for which transmission unit information is being requested.

Response fields:

in.EID: EID of the next bridge that is used to access the target endpoint, if any.

in.Message_Type: Message Type.

in.Incoming_TU_Size: Maximum supported incoming transmission unit size in increments of 16 bytes, starting from the baseline transmission unit size.

in.Outgoing_TU_Size: Maximum supported outgoing transmission unit size in increments of 16 bytes, starting from the baseline transmission unit.

5.5.2.13 Resolve UUID Support

The following values are valid for Resolve UUID messages only. Undefined for other messages. See Specification for field detailed descriptions.

Request fields:

in.UUID: Network ID bytes.

in.Entry_Handle: Entry Handle.

Response fields:

in.Next_Entry_Handle: Next Entry Handle.

in.Entries_Count: Number of EID entries being returned in this response.

in.EID: Array of Endpoint IDs.

in.Transport_Binding: Array of Physical Transport Binding Identifiers.

in.Media_Type_ID: Array of Physical Media Type Identifiers.

in.Address_Size: Array of Physical Address Sizes.

in.Address: Array of Physical Addresses.

5.5.3 PLDM Messages

The following values are valid for PLDM Messages only. Undefined for other messages.

in.Command_Code: PLDM Command Code field identifies the type of operation the message is requesting.

in.Completion_Code: The PLDM Completion Code field provides the status of the operation. This field is present for the PLDM Message Payload for PLDM response messages and is not present in PLDM request messages. Each command supports generic completion codes and may add additional codes:

Constant	Code	Generic Completion Code
_PLDM_COMPLETION_CODE_SUCCESS	0x00	SUCCESS
_PLDM_COMPLETION_CODE_ERROR	0x01	ERROR
_PLDM_COMPLETION_CODE_ERROR_INVALID_DATA	0x02	ERROR_INVALID_DATA
_PLDM_COMPLETION_CODE_ERROR_INVALID_LENGTH	0x03	ERROR_INVALID_LENGTH

_PLDM_COMPLETION_CODE_ERROR_NOT_READY	0x04	ERROR_NOT_READY
_PLDM_COMPLETION_CODE_ERROR_UNSUPPORTED_PLDM_CMD	0x05	ERROR_UNSUPPORTED_PLDM_CMD
_PLDM_COMPLETION_CODE_ERROR_INVALID_PLDM_TYPE	0x20	ERROR_INVALID_PLDM_TYPE
_PLDM_COMPLETION_CODE_COMMAND_SPECIFIC_FROM	0x80	COMMAND_SPECIFIC (begin of the range)
_PLDM_COMPLETION_CODE_COMMAND_SPECIFIC_TO	0xFF	COMMAND_SPECIFIC (end of the range)

in.D: Datagram bit. This bit is used to indicate whether the Instance ID field is being used for tracking and matching requests and responses, or just being used for asynchronous notifications.

in.Hdr_Ver: The Hdr Ver (Header Version) field identifies the header format.

in.IC: (MCTP integrity check bit) Indicates whether the MCTP message is covered by an overall MCTP message payload integrity check. This field is required to be the most significant bit of the first byte of the message body in the first packet of a message along with the message type bits.
0b = No MCTP message integrity check
1b = MCTP message integrity check is present

in.Inst_Id: The Instance ID (Instance Identifier) field is used to identify a new instance of a PLDM request to differentiate new PLDM requests that are sent to the same PLDM terminus.

in.Msg_Type: Defines the type of payload contained in the message data portion of the MCTP message. Is equal to _MCTP_MSG_PLDM (0x01) for each PLDM message.

in.PLDM_Type: The PLDM Type field identifies the type of PLDM that is being used in the control or data transfer carried out using this PLDM message.

Constant	Code	Type
_PLDM_TYPE_MSG_CTRL_DISC	0x00	PLDM Messaging Control and Discovery (DSP0240)
_PLDM_TYPE_SMBIOS	0x01	PLDM for SMBIOS (DSP0246)
_PLDM_TYPE_PL_MON_CONF	0x02	PLDM for Platform Monitoring and Control (DSP0248)
_PLDM_TYPE_BIOS_CTRL_CONFIG	0x03	PLDM for BIOS Control and Configuration (DSP0247)
_PLDM_TYPE_FRU	0x04	PLDM for FRU Data (DSP0257)
_PLDM_TYPE_FIRMWARE_UPDATE	0x05	PLDM for Firmware Update (DSP0267)
_PLDM_TYPE_RDE	0x06	PLDM for for Redfish Device Enablement (DSP0218)
_PLDM_TYPE_RESERVED_FROM	0x05	Reserved (begin of the range)
_PLDM_TYPE_RESERVED_TO	0x3E	Reserved (end of the range)
_PLDM_TYPE_OEM_SPECIFIC	0x3F	Reserved for OEM-specific PLDM commands

in.Rq: Request bit. This bit is used to help differentiate between PLDM request messages and other PLDM messages.

in.Sub_Type: D and Rq bit combinations:

Constant	Code	Type
_PLDM_SUB_TYPE_RESPONSE	0x00	PLDM response messages
_PLDM_SUB_TYPE_RESERVED	0x01	Reserved
_PLDM_SUB_TYPE_REQUEST	0x02	PLDM request messages
_PLDM_SUB_TYPE_UNACK_REQUEST_OR_ASYNC_NOTIFY	0x03	Unacknowledged PLDM request messages or asynchronous notifications

5.5.3.1 PLDM Messaging Control and Discovery messages

The following values are valid for PLDM Messaging Control and Discovery messages only. Undefined for other messages. See DSP0240 for detailed descriptions.

in.Alpha: Alpha part of Version

in.Command_Code: Command codes specific to the current PLDM type.

Constant	Code	Command
_PLDM_COMMAND_CODE_SET_TID	0x01	SetTID
_PLDM_COMMAND_CODE_GET_TID	0x02	GetTID
_PLDM_COMMAND_CODE_GET_PLDM_VERSION	0x03	GetPLDMVersion
_PLDM_COMMAND_CODE_GET_PLDM_TYPES	0x04	GetPLDMTypes
_PLDM_COMMAND_CODE_GET_PLDM_COMMANDS	0x05	GetPLDMCommands

in.Commands: PLDMCommands

in.Completion_Code: Completion codes specific to the current PLDM type.

Constant	Code	Completion Code
Generic codes are included		
_PLDM_COMPLETION_CODE_INVALID_DATA_TRANSFER_HANDLE	0x80	INVALID_DATA_TRANSFER_HANDLE
_PLDM_COMPLETION_CODE_INVALID_TRANSFER_OPERATION_FLAG	0x81	INVALID_TRANSFER_OPERATION_FLAG
_PLDM_COMPLETION_CODE_INVALID_PLDM_TYPE_IN_REQUEST_DATA	0x83	INVALID_PLDM_TYPE_IN_REQUEST_DATA
_PLDM_COMPLETION_CODE_INVALID_PLDM_VERSION_IN_REQUEST_DATA	0x84	INVALID_PLDM_VERSION_IN_REQUEST_DATA

in.Flag: TransferFlag, TransferOperationFlag

Constants for TransferFlag	Code	Flag
_PLDM_TRANSFER_FLAG_START	0x01	Start
_PLDM_TRANSFER_FLAG_MIDDLE	0x02	Middle
_PLDM_TRANSFER_FLAG_END	0x04	End
_PLDM_TRANSFER_FLAG_START_AND_END	0x05	StartAndEnd

Constants for TransferOperationFlag	Code	Flag
_PLDM_TRANSFER_OPERATION_FLAG_GET_NEXT_PART	0x00	GetNextPart
_PLDM_TRANSFER_OPERATION_FLAG_GET_FIRST_PART	0x01	GetFirstPart

in.Handle : DataTransferHandle, NextDataTransferHandle

in.Major: Major part of Version

in.Minor: Minor part of Version

in.TID : TID

in.Type : PLDMType

in.Types : PLDMTypes

in.Upd: Update part of Version

5.5.3.2 PLDM for SMBIOS messages

The following values are valid for PLDM for SMBIOS messages only. Undefined for other messages. See DSP0246 for detailed descriptions.

in.Checksum :. SMBIOSStructureTableIntegrityChecksum

in.Command_Code: Command codes specific to the current PLDM type.

Constant	Code	Command
_PLDM_COMMAND_CODE_GET_SMBIOS_STRUCTURE_TABLE_METADATA	0x01	GetSMBIOSStructureTableMetadata
_PLDM_COMMAND_CODE_SET_SMBIOS_STRUCTURE_TABLE_METADATA	0x02	SetSMBIOSStructureTableMetadata

_PLDM_COMMAND_CODE_GET_SMBIOS_STRUCTURE_TABLE	0x03	GetSMBIOSStructureTable
_PLDM_COMMAND_CODE_SET_SMBIOS_STRUCTURE_TABLE	0x04	SetSMBIOSStructureTable
_PLDM_COMMAND_CODE_GET_SMBIOS_STRUCTURE_BY_TYPE	0x05	GetSMBIOSStructureByType
_PLDM_COMMAND_CODE_GET_SMBIOS_STRUCTURE_BY_HANDLE	0x06	GetSMBIOSStructureByHandle

in.Completion_Code: Completion codes specific to the current PLDM type.

Constant	Code	Completion Code
Generic codes are included		
_PLDM_COMPLETION_CODE_INVALID_DATA_TRANSFER_HANDLE	0x80	INVALID_DATA_TRANSFER_HANDLE
_PLDM_COMPLETION_CODE_INVALID_TRANSFER_OPERATION_FLAG	0x81	INVALID_TRANSFER_OPERATION_FLAG
_PLDM_COMPLETION_CODE_INVALID_TRANSFER_FLAG	0x82	INVALID_TRANSFER_FLAG
_PLDM_COMPLETION_CODE_NO_SMBIOS_STRUCTURE_TABLE_METADATA	0x83	NO_SMBIOS_STRUCTURE_TABLE_METADATA
_PLDM_COMPLETION_CODE_INVALID_DATA_INTEGRITY_CHECK	0x84	INVALID_DATA_INTEGRITY_CHECK
_PLDM_COMPLETION_CODE_SMBIOS_STRUCTURE_TABLE_UNAVAILABLE	0x85	SMBIOS_STRUCTURE_TABLE_UNAVAILABLE
_PLDM_COMPLETION_CODE_NO_SMBIOS_STRUCTURES	0x86	NO_SMBIOS_STRUCTURES
_PLDM_COMPLETION_CODE_INVALID_SMBIOS_STRUCTURE_TYPE	0x87	INVALID_SMBIOS_STRUCTURE_TYPE
_PLDM_COMPLETION_CODE_INVALID_SMBIOS_STRUCTURE_HANDLE	0x88	INVALID_SMBIOS_STRUCTURE_HANDLE
_PLDM_COMPLETION_CODE_INVALID_SMBIOS_STRUCTURE_INSTANCE_ID	0x89	INVALID_SMBIOS_STRUCTURE_INSTANCE_ID

in.Flag: TransferOperationFlag, TransferFlag

Constants for TransferFlag	Code	Flag
_PLDM_TRANSFER_FLAG_START	0x01	Start
_PLDM_TRANSFER_FLAG_MIDDLE	0x02	Middle
_PLDM_TRANSFER_FLAG_END	0x04	End
_PLDM_TRANSFER_FLAG_START_AND_END	0x05	StartAndEnd

Constants for TransferOperationFlag	Code	Flag
_PLDM_TRANSFER_OPERATION_FLAG_GET_NEXT_PART	0x00	GetNextPart
_PLDM_TRANSFER_OPERATION_FLAG_GET_FIRST_PART	0x01	GetFirstPart

in.Handle : DataTransferHandle, NextDataTransferHandle, Handle

in.Length :. SMBIOSStructureTableLength

in.Major :. SMBIOSMajorVersion

in.Max_Size :. MaximumStructureSize

in.Minor :. SMBIOSMinorVersion

in.Struct_Numbers :. NumberOfSMBIOSStructures

in.Type :. Type

in.Struct_ID :. StructureInstanceID

5.5.3.3 PLDM for Platform Monitoring and Control messages

The following values are valid for PLDM for Platform Monitoring and Control messages only. Undefined for other messages. See DSP0248 for detailed descriptions.

5.5.3.3.1 Terminus commands

in.Addr_Info: eventReceiverAddress

in.Command_Code: Command codes specific to the current PLDM type.

Constant	Code	Command
_PLDM_COMMAND_CODE_SET_TID	0x01	SetTID
_PLDM_COMMAND_CODE_GET_TID	0x02	GetTID
_PLDM_COMMAND_CODE_GET_FRU_RECORD_BY_OPTION	0x03	GetTerminusUID
_PLDM_COMMAND_CODE_GET_TERMINUS_UID	0x04	SetEventReceiver
_PLDM_COMMAND_CODE_SET_EVENT_RECEIVER	0x05	GetEventReceiver
_PLDM_COMMAND_CODE_GET_EVENT_RECEIVER	0x0A	PlatformEventMessage

in.Completion_Code: Completion codes specific to the current command set.

Constant	Code	Completion Code
Generic codes are included		
_PLDM_COMPLETION_CODE_INVALID_PROTOCOL_TYPE	0x80	INVALID_PROTOCOL_TYPE
_PLDM_COMPLETION_CODE_UNSUPPORTED_EVENT_FORMAT_VERSION	0x81	UNSUPPORTED_EVENT_FORMAT_VERSION

in.Data_Size: sensorDataSize

Constant	Code	Data Size
_PLDM_PLMC_DATA_SIZE_UINT8	0x00	uint8
_PLDM_PLMC_DATA_SIZE_SINT8	0x01	sint8
_PLDM_PLMC_DATA_SIZE_UINT16	0x02	uint16
_PLDM_PLMC_DATA_SIZE_SINT16	0x03	sint16
_PLDM_PLMC_DATA_SIZE_UINT32	0x04	uint32
_PLDM_PLMC_DATA_SIZE_SINT32	0x05	sint32

in.Event_Class :. eventClass, effecterEventClass

Constant for effecterEventClass	Code	Event Class
_PLDM_PLMC_EFFECTER_EVENT_CLASS_EFFECTER_OP_STATE	0x00	effecterOpState

Constant for eventClass	Code	Event Class
_PLDM_PLMC_EVENT_CLASS_SENSOR_EVENT	0x00	sensorEvent
_PLDM_PLMC_EVENT_CLASS_EFFECTER_EVENT	0x01	effecterEvent

in.Event_Glob_Enable :. eventMessageGlobalEnable

Constant	Code	MessageEnable
_PLDM_PLMC_EVT_MSG_GLOBAL_DISABLE	0x00	disable
_PLDM_PLMC_EVT_MSG_GLOBAL_ENABLE	0x01	enable

in.Event_State :. eventState

in.Id :. effecterID

in.Offset :.sensorOffset

in.Pres_Op_State :. presentOpState

in.Pres_Reading :. presentReading

in.Prev_Event_State :. previousEventState

in.Prev_Op_State :. previousOpState

in.Sensor :.SensorID

in.Sensor_Event_Class :.SensorEventClass

Constant	Code	Completion Code
_PLDM_PLMC_SENSOR_EVENT_CLASS_SENSOR_OP_STATE		sensorOpState
_PLDM_PLMC_SENSOR_EVENT_CLASS_STATE_SENSOR_STATE		stateSensorState
_PLDM_PLMC_SENSOR_EVENT_CLASS_NUMERIC_SENSOR_STATE		numericSensorState

in.Status :. status

Constant	Code	Status
_PLDM_PLMC_EVENT_MESSAGE_STATUS_NO_LOGGING	0x00	noLogging
_PLDM_PLMC_EVENT_MESSAGE_STATUS_LOGGING_DISABLED	0x01	loggingDisabled
_PLDM_PLMC_EVENT_MESSAGE_STATUS_LOG_FULL	0x02	logFull
_PLDM_PLMC_EVENT_MESSAGE_STATUS_ACCEPTED_FOR_LOGGING	0x03	acceptedForLogging
_PLDM_PLMC_EVENT_MESSAGE_STATUS_LOGGED	0x04	logged
_PLDM_PLMC_EVENT_MESSAGE_STATUS_LOGGING_REJECTED	0x05	loggingRejected

in.TID :.TID

in.Transport :. transportProtocolType

Constant	Code	Protocol Type
_PLDM_TRANSPORT_PROTOCOL_TYPE_MCTP	0x00	MCTP
_PLDM_TRANSPORT_PROTOCOL_TYPE_VENDOR_SPECIFIC	0xFF	Vendor Specific

in.Version :.formatVersion

in.UUID_Value :.UUIDValue

5.5.3.3.2 PLDM Numeric Sensor commands

in.Command_Code: Command codes specific to the current PLDM type.

Constant	Code	Command
_PLDM_COMMAND_CODE_SET_NUMERIC_SENSOR_ENABLE	0x10	SetNumericSensorEnable
_PLDM_COMMAND_CODE_GET_SENSOR_READING	0x11	GetSensorReading
_PLDM_COMMAND_CODE_GET_SENSOR_THRESHOLDS	0x12	GetSensorThresholds
_PLDM_COMMAND_CODE_SET_SENSOR_THRESHOLDS	0x13	SetSensorThresholds
_PLDM_COMMAND_CODE_RESTORE_SENSOR_THRESHOLDS	0x14	RestoreSensorThresholds
_PLDM_COMMAND_CODE_GET_SENSOR_HYSTERESIS	0x15	GetSensorHysteresis
_PLDM_COMMAND_CODE_SET_SENSOR_HYSTERESIS	0x16	SetSensorHysteresis
_PLDM_COMMAND_CODE_INIT_NUMERIC_SENSOR	0x17	InitNumericSensor

in.Completion_Code: Completion codes specific to the current command set.

Constant	Code	Completion Code
Generic codes are included		
_PLDM_COMPLETION_CODE_INVALID_SENSOR_ID	0x80	INVALID_SENSOR_ID
SetNumericSensorEnable specific		
_PLDM_COMPLETION_CODE_INVALID_SENSOR_OPERATIONAL_STATE	0x81	INVALID_SENSOR_OPERATIONAL_STATE
_PLDM_COMPLETION_CODE_EVENT_GENERATION_NOT_SUPPORTED	0x82	EVENT_GENERATION_NOT_SUPPORTED
GetSensorReading specific		
_PLDM_COMPLETION_CODE_REARM_UNAVAILABLE_IN_PRESENT_STATE	0x81	REARM_UNAVAILABLE_IN_PRESENT_STATE

in.Data_Size: sensorDataSize

Constant	Code	Data Size

_PLDM_PLMC_DATA_SIZE_UINT8	0x00	uint8
_PLDM_PLMC_DATA_SIZE_SINT8	0x01	sint8
_PLDM_PLMC_DATA_SIZE_UINT16	0x02	uint16
_PLDM_PLMC_DATA_SIZE_SINT16	0x03	sint16
_PLDM_PLMC_DATA_SIZE_UINT32	0x04	uint32
_PLDM_PLMC_DATA_SIZE_SINT32	0x05	sint32

in.Event_Msg_Enable:. eventMsgEnable

Constant	Code	State
_PLDM_PLMC_EVENT_MESSAGE_ENABLE_ENABLE	0x00	enableEventMessages
_PLDM_PLMC_EVENT_MESSAGE_ENABLE_DISABLE	0x01	disableEventMessages
_PLDM_PLMC_EVENT_MESSAGE_ENABLE_NO_CHANGE	0xFF	noChange

in.Event_State:. rearmEventState, eventState. See constants for presentState

in.Hyst_Value:. hysteresis value

in.Lower_Thr_Crit:. lowerThresholdCritical

in.Lower_Thr_Fat :. lowerThresholdFatal

in.Lower_Thr_Warn :. lowerThresholdWarning

in.Message_Enable:. sensorEventMessageEnable

Constant for SetNumericSensorEnable command	Code	Message Enable
_PLDM_PLMC_EVENT_MESSAGE_NO_CHANGE	0x00	noChange
_PLDM_PLMC_EVENT_MESSAGE_DISABLE_EVENTS	0x01	disableEvents
_PLDM_PLMC_EVENT_MESSAGE_ENABLE_EVENTS	0x02	enableEvents
_PLDM_PLMC_EVENT_MESSAGE_ENABLE_OP_EVENTS_ONLY	0x03	enableOpEventsOnly
_PLDM_PLMC_EVENT_MESSAGE_ENABLE_STATE_EVENTS_ONLY	0x04	enableStateEventsOnly

Constant for GetSensorReading command	Code	Message Enable
_PLDM_PLMC_EVENT_MESSAGE_NO_EVENT_GENERATION	0x00	noEventGeneration
_PLDM_PLMC_EVENT_MESSAGE_EVENTS_DISABLED	0x01	eventsDisabled
_PLDM_PLMC_EVENT_MESSAGE_EVENTS_ENABLED	0x02	eventsEnabled
_PLDM_PLMC_EVENT_MESSAGE_OP_EVENTS_ONLY_ENABLED	0x03	opEventsOnlyEnabled
_PLDM_PLMC_EVENT_MESSAGE_STATE_EVENTS_ONLY_ENABLED	0x04	stateEventsOnlyEnabled

in.Pres_State :. presentState

Constant	Code	State
_PLDM_PLMC_SENSOR_STATE_UNKNOWN	0x00	Unknown
_PLDM_PLMC_SENSOR_STATE_NORMAL	0x01	Normal
_PLDM_PLMC_SENSOR_STATE_WARNING	0x02	Warning
_PLDM_PLMC_SENSOR_STATE_CRITICAL	0x03	Critical
_PLDM_PLMC_SENSOR_STATE_FATAL	0x04	Fatal
_PLDM_PLMC_SENSOR_STATE_LOWER_WARNING	0x05	LowerWarning
_PLDM_PLMC_SENSOR_STATE_LOWER_CRITICAL	0x06	LowerCritical
_PLDM_PLMC_SENSOR_STATE_LOWER_FATAL	0x07	LowerFatal
_PLDM_PLMC_SENSOR_STATE_UPPER_WARNING	0x08	UpperWarning
_PLDM_PLMC_SENSOR_STATE_UPPER_CRITICAL	0x09	UpperCritical
_PLDM_PLMC_SENSOR_STATE_UPPER_FATAL	0x0A	UpperFatal

in.Prev_State :. previousState. See constants for presentState

in.Sensor :.SensorID

in.Sensor_Op_State :. sensorOperationalState

Constant for GetSensorReading command	Code	State
_PLDM_PLMC_OP_STATE_GET_SENSOR_READING_ENABLED	0x00	enabled
_PLDM_PLMC_OP_STATE_GET_SENSOR_READING_DISABLED	0x01	disabled
_PLDM_PLMC_OP_STATE_GET_SENSOR_READING_UNAVAILABLE	0x02	unavailable
_PLDM_PLMC_OP_STATE_GET_SENSOR_READING_STATUS_UNKNOWN	0x03	statusUnknown
_PLDM_PLMC_OP_STATE_GET_SENSOR_READING_FAILED	0x04	failed
_PLDM_PLMC_OP_STATE_GET_SENSOR_READING_INITIALIZING	0x05	initializing
_PLDM_PLMC_OP_STATE_GET_SENSOR_READING_SHUTTING_DOWN	0x06	shuttingDown
_PLDM_PLMC_OP_STATE_GET_SENSOR_READING_IN_TEST	0x07	inTest

Constant for GetNumericEffecterValue command	Code	State
_PLDM_PLMC_OP_STATE_GET_NUMERIC_EFFECTER_VALUE_ENABLED_UPDATE_PENDING	0x00	enabled-updatePending
_PLDM_PLMC_OP_STATE_GET_NUMERIC_EFFECTER_VALUE_ENABLED_NO_UPDATE_PENDING	0x01	enablednoUpdatePending
_PLDM_PLMC_OP_STATE_GET_NUMERIC_EFFECTER_VALUE_DISABLED	0x02	disabled
_PLDM_PLMC_OP_STATE_GET_NUMERIC_EFFECTER_VALUE_UNAVAILABLE	0x03	unavailable
_PLDM_PLMC_OP_STATE_GET_NUMERIC_EFFECTER_VALUE_STATUS_UNKNOWN	0x04	statusUnknown
_PLDM_PLMC_OP_STATE_GET_NUMERIC_EFFECTER_VALUE_FAILED	0x05	failed
_PLDM_PLMC_OP_STATE_GET_NUMERIC_EFFECTER_VALUE_INITIALIZING	0x06	initializing
_PLDM_PLMC_OP_STATE_GET_NUMERIC_EFFECTER_VALUE_SHUTTING_DOWN	0x07	shuttingDown
_PLDM_PLMC_OP_STATE_GET_NUMERIC_EFFECTER_VALUE_IN_TEST	0x08	inTest

in.Set_Num_Reading:. setNumericReading

in.Upper_Thr_Crit:. upperThresholdCritical

in.Upper_Thr_Fat :. upperThresholdFatal

in.Upper_Thr_Warn :. upperThresholdWarning

5.5.3.3.3 PLDM State Sensor commands

in.Command_Code: Command codes specific to the current PLDM type.

Constant	Code	Command
_PLDM_COMMAND_CODE_SET_STATE_SENSOR_ENABLES	0x20	SetStateSensorEnables
_PLDM_COMMAND_CODE_GET_STATE_SENSOR_READINGS	0x21	GetStateSensorReadings
_PLDM_COMMAND_CODE_INIT_STATE_SENSOR	0x22	InitStateSensor

in.Completion_Code: Completion codes specific to the current command set.

Constant	Code	Completion Code
Generic codes are included		
_PLDM_COMPLETION_CODE_INVALID_SENSOR_ID	0x80	INVALID_SENSOR_ID
_PLDM_COMPLETION_CODE_UNSUPPORTED_SENSORSTATE_STATESENSOR	0x81	UNSUPPORTED_SENSORSTATE
_PLDM_COMPLETION_CODE_EVENT_GENERATION_NOT_SUPPORTED	0x82	EVENT_GENERATION_NOT_SUPPORTED

in.Event_Msg_Enable:. eventMsgEnable

Constant	Code	Message Enable
_PLDM_PLMC_EVENT_MESSAGE_ENABLE_ENABLE	0x00	enableEventMessages
_PLDM_PLMC_EVENT_MESSAGE_ENABLE_DISABLE	0x01	disableEventMessages
_PLDM_PLMC_EVENT_MESSAGE_ENABLE_NO_CHANGE	0xFF	noChange

in.Event_State:. eventState

in.Message_Enable :.eventMessageEnable

Constant	Code	Message Enable
_PLDM_PLMC_EVENT_MESSAGE_NO_CHANGE	0x00	noChange

_PLDM_PLMC_EVENT_MESSAGE_DISABLE_EVENTS	0x01	disableEvents
_PLDM_PLMC_EVENT_MESSAGE_ENABLE_EVENTS	0x02	enableEvents
_PLDM_PLMC_EVENT_MESSAGE_ENABLE_OP_EVENTS_ONLY	0x03	enableOpEventsOnly
_PLDM_PLMC_EVENT_MESSAGE_ENABLE_STATE_EVENTS_ONLY	0x04	enableStateEventsOnly

in.Pres_State :. presentState, sensorPresentState

Constant for presentState	Code	State
_PLDM_PLMC_SENSOR_STATE_UNKNOWN	0x00	Unknown
_PLDM_PLMC_SENSOR_STATE_NORMAL	0x01	Normal
_PLDM_PLMC_SENSOR_STATE_WARNING	0x02	Warning
_PLDM_PLMC_SENSOR_STATE_CRITICAL	0x03	Critical
_PLDM_PLMC_SENSOR_STATE_FATAL	0x04	Fatal
_PLDM_PLMC_SENSOR_STATE_LOWER_WARNING	0x05	LowerWarning
_PLDM_PLMC_SENSOR_STATE_LOWER_CRITICAL	0x06	LowerCritical
_PLDM_PLMC_SENSOR_STATE_LOWER_FATAL	0x07	LowerFatal
_PLDM_PLMC_SENSOR_STATE_UPPER_WARNING	0x08	UpperWarning
_PLDM_PLMC_SENSOR_STATE_UPPER_CRITICAL	0x09	UpperCritical
_PLDM_PLMC_SENSOR_STATE_UPPER_FATAL	0x0A	UpperFatal

in.Prev_State :. previousState. See constants for presentState.

in.Sensor :.SensorID

in.Sensor_Cnt :. compositeSensorCount

in.Sensor_Op_State :. sensorOperationalState

Constant	Code	State
_PLDM_PLMC_OP_STATE_GET_SENSOR_READING_ENABLED	0x00	enabled
_PLDM_PLMC_OP_STATE_GET_SENSOR_READING_DISABLED	0x01	disabled
_PLDM_PLMC_OP_STATE_GET_SENSOR_READING_UNAVAILABLE	0x02	unavailable

in.Sensor_Rearm:. sensorRearm

5.5.3.3.4 PLDM Effecter commands

in.Command_Code: Command codes specific to the current PLDM type.

Constant	Code	Command
_PLDM_COMMAND_CODE_SET_NUMERIC_EFFECTER_ENABLE	0x30	SetNumericEffecterEnable
_PLDM_COMMAND_CODE_SET_NUMERIC_EFFECTER_VALUE	0x31	SetNumericEffecterValue
_PLDM_COMMAND_CODE_GET_NUMERIC_EFFECTER_VALUE	0x32	GetNumericEffecterValue
_PLDM_COMMAND_CODE_SET_STATE_EFFECTER_ENABLES	0x38	SetStateEffecterEnables
_PLDM_COMMAND_CODE_SET_STATE_EFFECTER_STATES	0x39	SetStateEffecterStates
_PLDM_COMMAND_CODE_GET_STATE_EFFECTER_STATES	0x3A	GetStateEffecterStates

in.Completion_Code: Completion codes specific to the current command set.

Constant	Code	Completion Code
Generic codes are included		
_PLDM_COMPLETION_CODE_INVALID_EFFECTER_ID	0x80	INVALID_EFFECTER_ID
_PLDM_COMPLETION_CODE_INVALID_STATE_VALUE	0x81	INVALID_STATE_VALUE
_PLDM_COMPLETION_CODE_UNSUPPORTED_SENSORSTATE_EFFECTER	0x82	UNSUPPORTED_SENSORSTATE

in.Data_Size :.effecterDataSize

in.Eff_Cnt :.compositeEffecterCount

in.Eff_State :.effecterState

in.Eff_Value :.effecterValue

in.Event_Msg_Enable :. eventMsgEnable

Constant	Code	State
_PLDM_PLMC_EVENT_MESSAGE_SSEE_ENABLE_EVENTS	0x00	enableEvents
_PLDM_PLMC_EVENT_MESSAGE_SSEE_DISABLE_EVENTS	0x01	disableEvents
_PLDM_PLMC_EVENT_MESSAGE_SSEE_NO_CHANGE	0xFF	noChange

in.ID :.effecterID

in.Op_State :. effecterOperationalState

Constant for SetStateEffecterEnables command	Code	State
_PLDM_PLMC_EFFECTER_OPERATIONAL_STATE_SNEE_ENABLED	0x00	enabled
_PLDM_PLMC_EFFECTER_OPERATIONAL_STATE_SNEE_DISABLED	0x02	disabled
_PLDM_PLMC_EFFECTER_OPERATIONAL_STATE_SNEE_UNAVAILABLE	0x03	unavailable

Constant for GetNumericEffecterValue command	Code	State
_PLDM_PLMC_OP_STATE_GET_NUMERIC_EFFECTER_VALUE_ENABLED_UPDATE_PENDING	0x00	enabled-updatePending
_PLDM_PLMC_OP_STATE_GET_NUMERIC_EFFECTER_VALUE_ENABLED_NO_UPDATE_PENDING	0x01	enablednoUpdatePending
_PLDM_PLMC_OP_STATE_GET_NUMERIC_EFFECTER_VALUE_DISABLED	0x02	disabled
_PLDM_PLMC_OP_STATE_GET_NUMERIC_EFFECTER_VALUE_UNAVAILABLE	0x03	unavailable
_PLDM_PLMC_OP_STATE_GET_NUMERIC_EFFECTER_VALUE_STATUS_UNKNOWN	0x04	statusUnknown
_PLDM_PLMC_OP_STATE_GET_NUMERIC_EFFECTER_VALUE_FAILED	0x05	failed
_PLDM_PLMC_OP_STATE_GET_NUMERIC_EFFECTER_VALUE_INITIALIZING	0x06	initializing
_PLDM_PLMC_OP_STATE_GET_NUMERIC_EFFECTER_VALUE_SHUTTING_DOWN	0x07	shuttingDown
_PLDM_PLMC_OP_STATE_GET_NUMERIC_EFFECTER_VALUE_IN_TEST	0x08	inTest

in.Pend_Value :.pendingValue

in.Pend_State :.pendingState

in.Pres_State :.presentState

in.Pres_Value :.presentValue

in.Set_Request :.setRequest

Constant	Code	State
_PLDM_PLMC_SET_REQUEST_NO_CHANGE	0x00	noChange
_PLDM_PLMC_SET_REQUEST_REQUEST_SET	0x01	requestSet

5.5.3.3.5 PLDM Event Log commands

in.Add_100S :. mostRecentAddTimestamp100s

in.Add_Seconds :. mostRecentAddTimestampSeconds

in.Add_UTC_Offset :. mostRecentAddTimestampUTCOffset

in.Age :. configurableParameterSupport, [0] field

in.Age_Max :. AgeMax

in.Age_Min :. AgeMin

in.Clear_Policy :. activeLogClearingPolicy, logClearingPolicy, selectedLogClearingPolicy

Constant	Code	Completion Code
_PLDM_PLMC_CLEARING_POLICY_FILL_AND_STOP	0x00	fillAndStop
_PLDM_PLMC_CLEARING_POLICY_FIFO	0x01	FIFO
_PLDM_PLMC_CLEARING_POLICY_CLEAR_ON_AGE	0x02	clearOnAge

in.Cmp_100S :. compareTimestamp100s

in.Cmp_Seconds :. compareTimestampSeconds

in.Cmp_UTC_Offset :. compareTimestampUTCOffset

in.Command_Code: Command codes specific to the current PLDM type.

Constant	Code	Command
_PLDM_COMMAND_CODE_GET_PLDM_EVENT_LOG_INFO	0x40	GetPLDMEventLogInfo
_PLDM_COMMAND_CODE_ENABLE_PLDM_EVENT_LOGGING	0x41	EnablePLDMEventLogging
_PLDM_COMMAND_CODE_CLEAR_PLDM_EVENT_LOG	0x42	ClearPLDMEventLog
_PLDM_COMMAND_CODE_GET_PLDM_EVENT_LOG_TIMESTAMP	0x43	GetPLDMEventLogTimestamp
_PLDM_COMMAND_CODE_SET_PLDM_EVENT_LOG_TIMESTAMP	0x44	SetPLDMEventLogTimestamp
_PLDM_COMMAND_CODE_READ_PLDM_EVENT_LOG	0x45	ReadPLDMEventLog
_PLDM_COMMAND_CODE_GET_PLDM_EVENT_LOG_POLICY_INFO	0x46	GetPLDMEventLogPolicyInfo
_PLDM_COMMAND_CODE_SET_PLDM_EVENT_LOG_POLICY	0x47	SetPLDMEventLogPolicy
_PLDM_COMMAND_CODE_FIND_PLDM_EVENT_LOG_ENTRY	0x48	FindPLDMEventLogEntry

in.Completion_Code: Completion codes specific to the current command set.

Constant	Code	Completion Code
Generic codes are included		
_PLDM_COMPLETION_CODE_INVALID_SEARCH_TYPE	0x80	INVALID_SEARCH_TYPE
_PLDM_COMPLETION_CODE_INVALID_TRANSFER_OPERATION_FLAG	0x81	INVALID_TRANSFER_OPERATION_FLAG
_PLDM_COMPLETION_CODE_INVALID_ENTRY_ID	0x82	INVALID_ENTRY_ID

in.CRC :. transferCRC

in.Data :. transferredEntryData

in.Enable :. enableLogging

Constant	Code	Completion Code
_PLDM_PLMC_LOGGING_DISABLE_LOGGING	0x00	disableLogging
_PLDM_PLMC_LOGGING_ENABLE_LOGGING	0x01	enableLogging

in.Entry_100S :. entryTimestamp100s

in.Entry_Count :.entryCount

in.Entry_Seconds :. entryTimestampSeconds

in.Entry_UTC_Offset :. entryTimestampUTCOffset

in.Erase_100S :. mostRecentEraseTimestamp100s

in.Erase_Seconds :. mostRecentEraseTimestampSeconds

in.Erase_UTC_Offset :. mostRecentEraseTimestampUTCOffset

in.ID :. entryID

in.M :. M

in.M_Max :. MMax

in.M_Min :. MMin

in.M_Mpercent :. configurableParameterSupport, [1:2] field

in.Mpercent :. MPercentage

in.Mpercent_Max :. MPercentageMax

in.Mpercent_Min : MPercentageMin

in.N :. N

in.N_Max :. NMax

in.N_Min :. NMin

in.N_Npercent :. configurableParameterSupport, [3:4] field

in.Next_ID :. nextEntryID

in.Npercent :. NPercentage

in.Npercent_Max :. NPercentageMax

in.Npercent_Min :. NPercentageMin

in.Op_State :. logOperationalStatus

Constant	Code	Completion Code
_PLDM_PLMC_LOG_OPERATIONAL_STATUS_LOGGING_DISABLED	0x00	loggingDisabled
_PLDM_PLMC_LOG_OPERATIONAL_STATUS_ENABLED_READY	0x01	enabledReady
_PLDM_PLMC_LOG_OPERATIONAL_STATUS_CLEAR_IN_PROGRESS	0x02	clearInProgress
_PLDM_PLMC_LOG_OPERATIONAL_STATUS_ENABLED_FULL	0x03	enabledFull
_PLDM_PLMC_LOG_OPERATIONAL_STATUS_FAILED_LOGGING_DISABLED	0x04	failedLoggingDisabled
_PLDM_PLMC_LOG_OPERATIONAL_STATUS_FAILED_DISABLED	0x05	failedDisabled
_PLDM_PLMC_LOG_OPERATIONAL_STATUS_CORRUPTED	0x06	corrupted

in.Percent_Wear :. percentWear

in.Resolution :. timestampResolution

in.Search_Type :. searchType

Constant	Code	Type
_PLDM_PLMC_SEARCH_TYPE_NEWER_THAN	0x00	newerThan
_PLDM_PLMC_SEARCH_TYPE_OLDER_THAN	0x01	olderThan
_PLDM_PLMC_SEARCH_TYPE_OFFSET_FROM_START	0x02	offsetFromStart
_PLDM_PLMC_SEARCH_TYPE_OFFSET_FROM_END	0x03	offsetFromEnd

in.Set_Op :. setOperation

Constant	Code	Operation
_PLDM_PLMC_SET_OPERATION_CONFIGURE_ONLY	0x00	configureOnly
_PLDM_PLMC_SET_OPERATION_SET_ONLY	0x01	setOnly
_PLDM_PLMC_SET_OPERATION_CONFIGURE_AND_SET	0x02	configureAndSet

in.Size :. transferredDataSize

in.Split :. splitEntry

Constant	Code	Split
_PLDM_PLMC_SPLIT_ENTRY_FULL	0x00	full
_PLDM_PLMC_SPLIT_ENTRY_FIRST_FRAGMENT	0x01	firstFragment
_PLDM_PLMC_SPLIT_ENTRY_MIDDLE_FRAGMENT	0x02	middleFragment
_PLDM_PLMC_SPLIT_ENTRY_LAST_FRAGMENT	0x03	lastFragmen

in.Start_Point :. startingPoint

in.Storage_Used :. storagePercentUsed

in.Update_Event :. logUpdateEvent

Constant	Code	Event
_PLDM_PLMC_LOG_EVENT_NO_EVENT	0x00	noEvent
_PLDM_PLMC_LOG_EVENT_LOG_EVENT	0x01	logEvent

5.5.3.3.6 PDR Repository commands

Time resolution enumeration

Constant	Code	Resolution
_PLDM_TIME_RESOLUTION_MICROSECOND	0x00	microsecond
_PLDM_TIME_RESOLUTION_10MICROSECOND	0x01	10microsecond
_PLDM_TIME_RESOLUTION_100MICROSECOND	0x02	100microsecond
_PLDM_TIME_RESOLUTION_MILLISECOND	0x03	millisecond
_PLDM_TIME_RESOLUTION_10MILLISECOND	0x04	10millisecond
_PLDM_TIME_RESOLUTION_100MILLISECOND	0x05	100millisecond
_PLDM_TIME_RESOLUTION_SECOND	0x06	second
_PLDM_TIME_RESOLUTION_10SECOND	0x07	10second
_PLDM_TIME_RESOLUTION_MINUTE	0x08	minute
_PLDM_TIME_RESOLUTION_10MINUTE	0x09	10minute
_PLDM_TIME_RESOLUTION_HOUR	0x0A	hour
_PLDM_TIME_RESOLUTION_DAY	0x0B	day
_PLDM_TIME_RESOLUTION_MONTH	0x0C	month
_PLDM_TIME_RESOLUTION_YEAR	0x0D	year

in.Command_Code: Command codes specific to the current PLDM type.

Constant	Code	Command
_PLDM_COMMAND_CODE_GET_PDR_REPOSITORY_INFO	0x50	GetPDRRepositoryInfo
_PLDM_COMMAND_CODE_GET_PDR	0x51	GetPDR
_PLDM_COMMAND_CODE_FIND_PDR	0x52	FindPDR
_PLDM_COMMAND_CODE_RUN_INIT_AGENT	0x58	RunInitAgent

in.Completion_Code: Completion codes specific to the current command set.

Constant for GetPDR command	Code	Completion Code
Generic codes are included		
_PLDM_COMPLETION_CODE_INVALID_DATA_TRANSFER_HANDLE	0x80	INVALID_DATA_TRANSFER_HANDLE

_PLDM_COMPLETION_CODE_INVALID_TRANSFER_OPERATION_FLAG	0x81	INVALID_TRANSFER_OPERATION_FLAG
_PLDM_COMPLETION_CODE_INVALID_RECORD_HANDLE	0x82	INVALID_RECORD_HANDLE
_PLDM_COMPLETION_CODE_INVALID_RECORD_CHANGE_NUMBER	0x83	INVALID_RECORD_CHANGE_NUMBER
_PLDM_COMPLETION_CODE_TRANSFER_TIMEOUT	0x84	TRANSFER_TIMEOUT
_PLDM_COMPLETION_CODE_REPOSITORY_UPDATE_IN_PROGRESS	0x85	REPOSITORY_UPDATE_IN_PROGRESS

Constant for FindPDR command	Code	Completion Code
Generic codes are included		
_PLDM_COMPLETION_CODE_INVALID_FIND_HANDLE	0x80	INVALID_FIND_HANDLE
_PLDM_COMPLETION_CODE_INVALID_FIND_OPERATION_FLAG	0x81	INVALID_FIND_OPERATION_FLAG
_PLDM_COMPLETION_CODE_INVALID_PDR_TYPE	0x82	INVALID_PDR_TYPE
_PLDM_COMPLETION_CODE_INVALID_PARAMETER_FORMAT_NUMBER	0x83	INVALID_PARAMETER_FORMAT_NUMBER
_PLDM_COMPLETION_CODE_INVALID_FIND_PARAMETERS	0x84	INVALID_FIND_PARAMETERS
_PLDM_COMPLETION_CODE_REPOSITORY_UPDATE_IN_PROGRESS	0x85	REPOSITORY_UPDATE_IN_PROGRESS

in.CRC :. transferCRC

in.Data :. recordData

in.Data_Transfer_Timeout :. dataTransferHandleTimeout

in.Init_Cond_Emul :. initConditionEmulation

Constants for TransferOperationFlag	Code	Flag
_PLDM_PLMC_INIT_COND_EMUL_INITIALIZATION_AGENT_RESTART	0x00	InitializationAgentRestart
_PLDM_PLMC_INIT_COND_EMUL_PLDM_SUBSYSTEM_POWER_UP	0x01	PLDMSubsystemPowerUp
_PLDM_PLMC_INIT_COND_EMUL_SYSTEM_HARD_RESET	0x02	SystemHardReset
_PLDM_PLMC_INIT_COND_EMUL_SYSTEM_WARM_RESET	0x03	SystemWarmReset
_PLDM_PLMC_INIT_COND_EMUL_PLDM_TERMINUS_ONLINE	0x04	PLDMTerminusOnline

in.Flag :. transferFlag, transferOperationFlag

Constants for TransferOperationFlag	Code	Flag
_PLDM_TRANSFER_OPERATION_FLAG_GET_NEXT_PART	0x00	GetNextPart
_PLDM_TRANSFER_OPERATION_FLAG_GET_FIRST_PART	0x01	GetFirstPart

Constants for TransferFlag	Code	Flag
_PLDM_TRANSFER_FLAG_START	0x01	Start
_PLDM_TRANSFER_FLAG_MIDDLE	0x02	Middle
_PLDM_TRANSFER_FLAG_END	0x04	End
_PLDM_TRANSFER_FLAG_START_AND_END	0x05	StartAndEnd

in.Format_Number :. parameterFormatNumber

in.Handle :. dataTransferHandle, nextDataTransferHandle, findHandle

in.Init_Cond_Emul :. initConditionEmulation

in.Largest_Rec_Size :. largestRecordSize

in.Next_Handle:. nextFindHandle

in.Next_Rec_Handle:. nextRecordHandle

in.OEM_Upd_Day :. OEMUpdateTime, day within the month

in.OEM_Upd_Hour :. OEMUpdateTime, hour within the day

in.OEM_Upd_Min :. OEMUpdateTime, minute within the hour

in.OEM_Upd_Month :. OEMUpdateTime, month

in.OEM_Upd_Sec :. OEMUpdateTime, seconds within the minute

in.OEM_Upd_Time_Res :. OEMUpdateTime, Time resolution. See Time resolution enumeration

in.OEM_Upd_uSec :. OEMUpdateTime, microsecond within the second

in.OEM_Upd_UTC_Off_Min :. OEMUpdateTime, UTC offset in minutes

in.OEM_Upd_UTC_Res :. OEMUpdateTime, UTCresolution

Constant	Code	Resolution
_PLDM_UTC_RESOLUTION_UNSPECIFIED	0x00	UTCunspecified
_PLDM_UTC_RESOLUTION_MINUTE	0x01	minute
_PLDM_UTC_RESOLUTION_10MINUTE	0x02	10minute
_PLDM_UTC_RESOLUTION_HOUR	0x03	hour

in.OEM_Upd_Year:. OEMUpdateTime, year

in.Op_Flag:. OEMUpdateTime, year

Constant	Code	Flag
_PLDM_UTC_RESOLUTION_UNSPECIFIED	0x00	findNext
_PLDM_UTC_RESOLUTION_MINUTE	0x01	findFirst

in.PDR_Type:. PDRType

in.Rec_Change_Number:. recordChangeNumber

in.Rec_Count:. recordCount

in.Rec_Handle:. recordHandle

in.Rep_Size:. repositorySize

in.Req_Count:. requestCount

in.Response_Count:. responseCount

in.State :.repositoryState

Constant	Code	Command
_PLDM_PLMC_AVAILABLE	0x00	available
_PLDM_PLMC_UPDATE_IN_PROGRESS	0x01	updateInProgress
_PLDM_PLMC_FAILED	0x02	failed

in.TID :.TID

in.Upd_Day :. updateTime, day within the month

in.Upd_Hour :. updateTime, hour within the day

in.Upd_Min :. updateTime, minute within the hour

in.Upd_Month :. updateTime, month

in.Upd_Sec :. updateTime, seconds within the minute

in.Upd_Time_Res :. updateTime, Time resolution. See Time resolution enumeration

in.Upd_uSec :. updateTime, microsecond within the second

in.Upd_UTC_Off_Min :. updateTime, UTC offset in minutes

in.Upd_UTC_Res :. updateTime, UTCresolution

Constant	Code	Resolution
_PLDM_UTC_RESOLUTION_UNSPECIFIED	0x00	UTCunspecified
_PLDM_UTC_RESOLUTION_MINUTE	0x01	minute
_PLDM_UTC_RESOLUTION_10MINUTE	0x02	10minute
_PLDM_UTC_RESOLUTION_HOUR	0x03	hour

in.Upd_Year:. updateTime, year

in.Wildcards :. wildcards

5.5.3.3.7 FindPDR Command Parameters

rateUnit enumeration:

Constant	Code	Rate Unit
_PLDM_PLMC_RATE_UNIT_NONE	0x00	None
_PLDM_PLMC_RATE_UNIT_MICROSECOND	0x01	Per MicroSecond
_PLDM_PLMC_RATE_UNIT_MILLISECOND	0x02	Per MilliSecond
_PLDM_PLMC_RATE_UNIT_SECOND	0x03	Per Second
_PLDM_PLMC_RATE_UNIT_MINUTE	0x04	Per Minute
_PLDM_PLMC_RATE_UNIT_HOUR	0x05	Per Hour
_PLDM_PLMC_RATE_UNIT_DAY	0x06	Per Day
_PLDM_PLMC_RATE_UNIT_WEEK	0x07	Per Week
_PLDM_PLMC_RATE_UNIT_MONTH	0x08	Per Month
_PLDM_PLMC_RATE_UNIT_YEAR	0x09	Per Year

sensorUnits enumeration:

Constant	Co-de	Rate Unit	Constant	Co-de	Rate Unit
_PLDM_PLMC_UNIT_NONE	0	None	_PLDM_PLMC_UNIT_OUNCE_INCHES	44	Ounce-Inches
_PLDM_PLMC_UNIT_UNSPECIFIED	1	Unspecified	_PLDM_PLMC_UNIT_GAUSS	45	Gauss
_PLDM_PLMC_UNIT_DEGREES_C	2	Degrees C	_PLDM_PLMC_UNIT_GILBERTS	46	Gilberts
_PLDM_PLMC_UNIT_DEGREES_F	3	Degrees F	_PLDM_PLMC_UNIT_HENRIES	47	Henries
_PLDM_PLMC_UNIT_DEGREES_K	4	Degrees K	_PLDM_PLMC_UNIT_FARADS	48	Farads
_PLDM_PLMC_UNIT_VOLTS	5	Volts	_PLDM_PLMC_UNIT_OHMS	49	Ohms
_PLDM_PLMC_UNIT_AMPS	6	Amps	_PLDM_PLMC_UNIT_SIEMENS	50	Siemens
_PLDM_PLMC_UNIT_WATTS	7	Watts	_PLDM_PLMC_UNIT_MOLES	51	Moles
_PLDM_PLMC_UNIT_JOULES	8	Joules	_PLDM_PLMC_UNIT_BECQUERELS	52	Becquerels
_PLDM_PLMC_UNIT_COULOMBS	9	Coulombs	_PLDM_PLMC_UNIT_PPM	53	PPM (parts/million)
_PLDM_PLMC_UNIT_VA	10	VA	_PLDM_PLMC_UNIT_DECIBELS	54	Decibels
_PLDM_PLMC_UNIT_NITS	11	Nits	_PLDM_PLMC_UNIT_DBA	55	DbA
_PLDM_PLMC_UNIT_LUMENS	12	Lumens	_PLDM_PLMC_UNIT_DBC	56	DbC
_PLDM_PLMC_UNIT_LUX	13	Lux	_PLDM_PLMC_UNIT_GRAYS	57	Grays
_PLDM_PLMC_UNIT_CANDELAS	14	Candelas	_PLDM_PLMC_UNIT_SIEVERTS	58	Sieverts
_PLDM_PLMC_UNIT_KPA	15	kPa	_PLDM_PLMC_UNIT_COLOR_TEMPERATURE_DEGREES_K	59	Color Temperature Degrees K
_PLDM_PLMC_UNIT_PSI	16	PSI	_PLDM_PLMC_UNIT_BITS	60	Bits
_PLDM_PLMC_UNIT_NEWTONS	17	Newtons	_PLDM_PLMC_UNIT_BYTES	61	Bytes
_PLDM_PLMC_UNIT_CFM	18	CFM	_PLDM_PLMC_UNIT_WORDS	62	Words (data)
_PLDM_PLMC_UNIT_RPM	19	RPM	_PLDM_PLMC_UNIT_DOUBLE_WORDS	63	DoubleWords

_PLDM_PLMC_UNIT_HERTZ	20	Hertz	_PLDM_PLMC_UNIT_QUAD_WORDS	64	QuadWords
_PLDM_PLMC_UNIT_SECONDS	21	Seconds	_PLDM_PLMC_UNIT_PERCENTAGE	65	Percentage
_PLDM_PLMC_UNIT_MINUTES	22	Minutes	_PLDM_PLMC_UNIT_PASCALS	66	Pascals
_PLDM_PLMC_UNIT_HOURS	23	Hours	_PLDM_PLMC_UNIT_COUNTS	67	Counts
_PLDM_PLMC_UNIT_DAYS	24	Days	_PLDM_PLMC_UNIT_GRAMS	68	Grams
_PLDM_PLMC_UNIT_WEEKS	25	Weeks	_PLDM_PLMC_UNIT_NEWTON_METERS	69	Newton-meters
_PLDM_PLMC_UNIT_MILS	26	Mils	_PLDM_PLMC_UNIT_HITS	70	Hits
_PLDM_PLMC_UNIT_INCHES	27	Inches	_PLDM_PLMC_UNIT_MISSES	71	Misses
_PLDM_PLMC_UNIT_FEET	28	Feet	_PLDM_PLMC_UNIT_RETRIES	72	Retries
_PLDM_PLMC_UNIT_CUBIC_INCHES	29	Cubic Inches	_PLDM_PLMC_UNIT_OVERRUNS_OVERFLOWS	73	Overruns/Overflows
_PLDM_PLMC_UNIT_CUBIC_FEET	30	Cubic Feet	_PLDM_PLMC_UNIT_UNDERRUNS	74	Underruns
_PLDM_PLMC_UNIT_METERS	31	Meters	_PLDM_PLMC_UNIT_COLLISIONS	75	Collisions
_PLDM_PLMC_UNIT_CUBIC_CENTIMETERS	32	Cubic Centimeters	_PLDM_PLMC_UNIT_PACKETS	76	Packets
_PLDM_PLMC_UNIT_CUBIC_METERS	33	Cubic Meters	_PLDM_PLMC_UNIT_MESSAGES	77	Messages
_PLDM_PLMC_UNIT_LITERS	34	Liters	_PLDM_PLMC_UNIT_CHARACTERS	78	Characters
_PLDM_PLMC_UNIT_FLUID_OUNCES	35	Fluid Ounces	_PLDM_PLMC_UNIT_ERRORS	79	Errors
_PLDM_PLMC_UNIT_RADIANS	36	Radians	_PLDM_PLMC_UNIT_CORRECTED_ERRORS	80	Corrected Errors
_PLDM_PLMC_UNIT_STERADIANS	37	Steradians	_PLDM_PLMC_UNIT_UNCORRECTABLE_ERRORS	81	Uncorrectable Errors
_PLDM_PLMC_UNIT_REVOLUTIONS	38	Revolutions	_PLDM_PLMC_UNIT_SQUARE_MILS	82	Square Mils
_PLDM_PLMC_UNIT_CYCLES	39	Cycles	_PLDM_PLMC_UNIT_SQUARE_INCHES	83	Square Inches
_PLDM_PLMC_UNIT_GRAVITIES	40	Gravities	_PLDM_PLMC_UNIT_SQUARE_FEET	84	Square Feet
_PLDM_PLMC_UNIT_OUNCES	41	Ounces	_PLDM_PLMC_UNIT_SQUARE_CENTIMETERS	85	Square Centimeters
_PLDM_PLMC_UNIT_POUNDS	42	Pounds	_PLDM_PLMC_UNIT_SQUARE_METERS	86	Square Meters
_PLDM_PLMC_UNIT_FOOT_POUNDS	43	Foot-Pounds	_PLDM_PLMC_UNIT_OEM_UNIT	255	OEMUnit

in.Association_Type :.associationType

Constant	Code	Resolution
_PLDM_PLMC_ASSOCIATION_TYPE_PHYSICAL_TO_PHYSICAL_CONTAINMENT	0x00	physicalToPhysicalContainment
_PLDM_PLMC_ASSOCIATION_TYPE_LOGICAL_CONTAINMENT	0x01	logicalContainment

in.Aux_OEM_Unit_Handle :.auxOEMUnitHandle

in.Aux_Unit :.auxUnit. See sensorUnits enumeration for constants list.

in.Aux_Unit_Modifier :.auxUnitModifier

in.Auxrate_Unit :.auxrateUnit. See rateUnit enumeration for constants list.

in.Base_OEM_Unit_Handle :.baseOEMUnitHandle

in.Base_Unit :.baseUnit. See sensorUnits enumeration for constants list.

in.Container_Ent_Cont_ID :.containerEntityContainerID

in.Container_Ent_Inst_Num :.containerEntityInstanceNumber

in.Container_Ent_Type :.containerEntityType

in.Container_ID :.containerID

in.Ent_Instance_Number :.entityInstanceNumber

in.Ent_Type :.entityType

in.ID :.effecterID

in.Interrupt_Source_Ent_Container_ID :.interruptSourceEntityContainerID

in.Interrupt_Source_Ent_Inst_Num :. interruptSourceEntityInstanceNumber

in.Interrupt_Source_Ent_Type :. interruptSourceEntityType

in.Interrupt_Target_Ent_Container_ID :.interruptTargetEntityInstanceNumber

in.Interrupt_Target_Ent_Inst_Num :.interruptTargetEntityInstanceNumber

in.Interrupt_Target_Ent_Type :.interruptTargetEntityType

in.OEM_Ent_ID_Handle :.OEMSEntityIDHandle, [14:0]

in.OEM_Record_ID :.OEMRecordID

in.OEM_State_Set_ID :.OEMStateSetID

in.OEM_State_Set_ID_Handle :.OEMStateSetIDHandle

in.OEM_Unit_Handle :.OEMUnitHandle

in.OEM_Unit_ID :.OEMUnitID

in.Rate_Unit :.rateUnit. See rateUnit enumeration for constants list.

in.Sensor :.sensorID

in.State_Set_ID :.stateSetID

in.Terminus_Handle :. PLDMTerminusHandle

in.TID :.TID

in.Unit_Modifier :.unitModifier

in.Vendor_IANA :.vendorIANA

5.5.3.4 PLDM for BIOS Control and Configuration messages

The following values are valid for PLDM for BIOS Control and Configuration messages only. Undefined for other messages. See DSP0247 for detailed descriptions.

in.Command_Code: Command codes specific to the current PLDM type.

Constant	Code	Command
_PLDM_COMMAND_CODE_GET_BIOS_TABLE	0x01	GetBIOSTable
_PLDM_COMMAND_CODE_SET_BIOS_TABLE	0x02	SetBIOSTable
_PLDM_COMMAND_CODE_UPDATE_BIOS_TABLE	0x03	UpdateBIOSTable
_PLDM_COMMAND_CODE_GET_BIOS_TABLE_TAGS	0x04	GetBIOSTableTags
_PLDM_COMMAND_CODE_SET_BIOS_TABLE_TAGS	0x05	SetBIOSTableTags
_PLDM_COMMAND_CODE_ACCEPT_BIOS_ATTRIBUTES_PENDINGVALUES	0x06	AcceptBIOSAttributesPendingValues
_PLDM_COMMAND_CODE_SET_BIOS_ATTRIBUTE_CURRENT_VALUE	0x07	SetBIOSAttributeCurrentValue
_PLDM_COMMAND_CODE_GET_BIOS_ATTRIBUTE_CURRENT_VALUE_BY_HANDLE	0x08	GetBIOSAttributeCurrentValueByHandle
_PLDM_COMMAND_CODE_GET_BIOS_ATTRIBUTE_PENDING_VALUE_BY_HANDLE	0x09	GetBIOSAttributePendingValueByHandle
_PLDM_COMMAND_CODE_GET_BIOS_ATTRIBUTE_CURRENT_VALUE_BY_TYPE	0x0A	GetBIOSAttributeCurrentValueByType
_PLDM_COMMAND_CODE_GET_BIOS_ATTRIBUTE_PENDING_VALUE_BY_TYPE	0x0B	GetBIOSAttributePendingValueByType
_PLDM_COMMAND_CODE_GET_DATE_TIME	0x0C	GetDateTime
_PLDM_COMMAND_CODE_SET_DATE_TIME	0x0D	SetDateTime
_PLDM_COMMAND_CODE_GET_BIOS_STRING_TABLE_STRING_TYPE	0x0E	GetBIOSStringTableStringType
_PLDM_COMMAND_CODE_SET_BIOS_STRING_TABLE_STRING_TYPE	0x0F	SetBIOSStringTableStringType

in.Completion_Code: Completion codes specific to the current PLDM type.

Constant	Code	Completion Code
Generic codes are included		
_PLDM_COMPLETION_CODE_INVALID_DATA_TRANSFER_HANDLE	0x80	INVALID_DATA_TRANSFER_HANDLE
_PLDM_COMPLETION_CODE_INVALID_TRANSFER_OPERATION_FLAG	0x81	INVALID_TRANSFER_OPERATION_FLAG
_PLDM_COMPLETION_CODE_INVALID_TRANSFER_FLAG	0x82	INVALID_TRANSFER_FLAG
_PLDM_COMPLETION_CODE_BIOS_TABLE_UNAVAILABLE	0x83	BIOS_TABLE_UNAVAILABLE
_PLDM_COMPLETION_CODE_INVALID_BIOS_TABLE_DATA_INTEGRITY_CHECK	0x84	INVALID_BIOS_TABLE_DATA_INTEGRITY_CHECK
_PLDM_COMPLETION_CODE_INVALID_BIOS_TABLE_TYPE	0x85	INVALID_BIOS_TABLE_TYPE
_PLDM_COMPLETION_CODE_BIOS_TABLE_TAG_UNAVAILABLE	0x86	BIOS_TABLE_TAG_UNAVAILABLE
_PLDM_COMPLETION_CODE_INVALID_BIOS_TABLE_TAG_TYPE	0x87	INVALID_BIOS_TABLE_TAG_TYPE
_PLDM_COMPLETION_CODE_INVALID_BIOS_ATTR_HANDLE	0x88	INVALID_BIOS_ATTR_HANDLE
_PLDM_COMPLETION_CODE_INVALID_BIOS_ATTR_TYPE	0x89	INVALID_BIOS_ATTR_TYPE

in.Attrib_Type :. AttributeType

Constants for AttributeType	Code	Type
_PLDM_BIOS_ATTRIB_TYPE_BIOS_ENUMERATION	0x00	BIOSEnumeration
_PLDM_BIOS_ATTRIB_TYPE_BIOS_STRING	0x01	BIOSString
_PLDM_BIOS_ATTRIB_TYPE_BIOS_PASSWORD	0x02	BIOSPassword
_PLDM_BIOS_ATTRIB_TYPE_BIOS_INTEGER	0x03	BIOSInteger
_PLDM_BIOS_ATTRIB_TYPE_BIOS_BOOT_CONFIG_SETTING	0x04	BIOSBootConfigSetting
_PLDM_BIOS_ATTRIB_TYPE_BIOS_COLLECTION	0x05	BIOSCollection
_PLDM_BIOS_ATTRIB_TYPE_BIOS_CONFIG_SET	0x06	BIOSConfigSet
_PLDM_BIOS_ATTRIB_TYPE_BIOS_ENUMERATION_READONLY	0x80	BIOSEnumerationReadOnly
_PLDM_BIOS_ATTRIB_TYPE_BIOS_STRING_READ_ONLY	0x81	BIOSStringReadOnly
_PLDM_BIOS_ATTRIB_TYPE_BIOS_PASSWORD_READ_ONLY	0x82	BIOSPasswordReadOnly
_PLDM_BIOS_ATTRIB_TYPE_BIOS_INTEGER_READ_ONLY	0x83	BIOSIntegerReadOnly
_PLDM_BIOS_ATTRIB_TYPE_BIOS_BOOT_CONFIG_SETTING_READ_ONLY	0x84	BIOSBootConfigSettingReadOnly
_PLDM_BIOS_ATTRIB_TYPE_BIOS_COLLECTION_READ_ONLY	0x85	BIOSCollectionReadOnly
_PLDM_BIOS_ATTRIB_TYPE_BIOS_CONFIG_SET_READ_ONLY	0x86	BIOSConfigSetReadOnly

in.Day_Month :.Day of the Month

in.Flag :. TransferFlag, ransferOperationFlag

Constants for TransferFlag	Code	Flag
_PLDM_TRANSFER_FLAG_START	0x01	Start
_PLDM_TRANSFER_FLAG_MIDDLE	0x02	Middle
_PLDM_TRANSFER_FLAG_END	0x04	End
_PLDM_TRANSFER_FLAG_START_AND_END	0x05	StartAndEnd

Constants for TransferOperationFlag	Code	Flag
_PLDM_TRANSFER_OPERATION_FLAG_GET_NEXT_PART	0x00	GetNextPart
_PLDM_TRANSFER_OPERATION_FLAG_GET_FIRST_PART	0x01	GetFirstPart

in.Handle :. DataTransferHandle, NextDataTransferHandle, AttributeHandle, BIOSAttributeHandle

in.Hour :.Hours

in.Min :.Minutes

in.Month :.Month

in.Number :. NumberOfTables

in.Table_Tag :. TableTag

in.Table_Type :. TableType

Constants for TableType	Code	Completion Code
_PLDM_BIOS_TABLE_TYPE_BIOS_STRING_TABLE	0x00	BIOSStringTable

_PLDM_BIOS_TABLE_TYPE_BIOS_ATTRIBUTE_TABLE	0x01	BIOSAttributeTable
_PLDM_BIOS_TABLE_TYPE_BIOS_ATTRIBUTE_VALUE_TABLE	0x02	BIOSAttributeValueTable
_PLDM_BIOS_TABLE_TYPE_BIOS_ATTRIBUTE_PENDING_VALUE_TABLE	0x03	BIOSAttributePendingValueTable

in.Sec :.Seconds

in.String_Type :.StringType

Constants for StringType	Code	Type
_PLDM_BIOS_STRING_TYPE_UNKNOWN	0x00	Unknown
_PLDM_BIOS_STRING_TYPE_ASCII	0x01	ASCII
_PLDM_BIOS_STRING_TYPE_HEX	0x02	Hex
_PLDM_BIOS_STRING_TYPE_UTF_8	0x03	UTF-8
_PLDM_BIOS_STRING_TYPE_UTF_16LE	0x04	UTF-16LE
_PLDM_BIOS_STRING_TYPE_UTF_16BE	0x05	UTF-16BE
_PLDM_BIOS_STRING_TYPE_VENDOR_SPECIFIC	0xFF	Vendor Specific

in.Year :.Year

5.5.3.5 PLDM for FRU Data messages

The following values are valid for PLDM for FRU messages only. Undefined for other messages. See DSP0257 for detailed descriptions.

in.Command_Code: Command codes specific to the current PLDM type.

Constant	Code	Command
_PLDM_COMMAND_CODE_GET_FRU_RECORD_TABLE_METADATA	0x01	GetFRURecordTableMetadata
_PLDM_COMMAND_CODE_GET_FRU_RECORD_TABLE	0x02	GetFRURecordTable
_PLDM_COMMAND_CODE_SET_FRU_RECORD_TABLE	0x03	SetFRURecordTable
_PLDM_COMMAND_CODE_GET_FRU_RECORD_BY_OPTION	0x04	GetFRURecordByOption

in.Completion_Code: Completion codes specific to the current PLDM type.

Constant	Code	Completion Code
Generic codes are included		
_PLDM_COMPLETION_CODE_INVALID_DATA_TRANSFER_HANDLE	0x80	INVALID_DATA_TRANSFER_HANDLE
_PLDM_COMPLETION_CODE_INVALID_TRANSFER_OPERATION_FLAG	0x81	INVALID_TRANSFER_OPERATION_FLAG
_PLDM_COMPLETION_CODE_INVALID_TRANSFER_FLAG	0x82	INVALID_TRANSFER_FLAG
_PLDM_COMPLETION_CODE_NO_FRU_DATA_STRUCTURE_TABLE_METADATA	0x83	NO_FRU_DATA_STRUCTURE_TABLE_METADATA
_PLDM_COMPLETION_CODE_INVALID_DATA_INTEGRITY_CHECK	0x84	INVALID_DATA_INTEGRITY_CHECK
_PLDM_COMPLETION_CODE_FRU_DATA_STRUCTURE_TABLE_UNAVAILABLE	0x85	FRU_DATA_STRUCTURE_TABLE_UNAVAILABLE

in.Checksum :. FRU DATAStructureTableIntegrityChecksum

in.Field_Type :. Field Type

in.Flag :. TransferFlag, TransferOperationFlag

Constants for TransferFlag	Code	Flag
_PLDM_TRANSFER_FLAG_START	0x01	Start
_PLDM_TRANSFER_FLAG_MIDDLE	0x02	Middle
_PLDM_TRANSFER_FLAG_END	0x04	End
_PLDM_TRANSFER_FLAG_START_AND_END	0x05	StartAndEnd

Constants for TransferOperationFlag	Code	Flag
_PLDM_TRANSFER_OPERATION_FLAG_GET_NEXT_PART	0x00	GetNextPart
_PLDM_TRANSFER_OPERATION_FLAG_GET_FIRST_PART	0x01	GetFirstPart

in.Id_Number :. Total number of Record Set Identifiers in table

in.Handle :. DataTransferHandle, NextDataTransferHandle

in.Length :. FRUTableLength

in.Major :. FRUDATAMajorVersion

in.Max_Size :. FRUTableMaximumSize

in.Minor :. FRUDATAMinorVersion

in.Record_Set_Id :. Record Set Identifier

in.Record_Type :. Record Type

in.Records_Number :. Total number of records in table

in.Table_Handle :. FRUTableHandle

5.5.3.6 PLDM for Firmware Update messages

Will be filled.

5.5.3.7 PLDM for Redfish Device Enablement messages

The following values are valid for Redfish Device Enablement messages only. Undefined for other messages. See DSP0218 for detailed descriptions.

in.Command_Code: Command codes specific to the current PLDM type.

Constant	Code	Command
_PLDM_COMMAND_CODE_NEGOTIATE_RDE_PARAMETERS	0x01	NegotiateRedfishParameters
_PLDM_COMMAND_CODE_NEGOTIATE_MEDIUM_PARAMETERS	0x02	NegotiateMediumParameters
_PLDM_COMMAND_CODE_GET_SCHEMA_DICTIONARY	0x03	GetSchemaDictionary
_PLDM_COMMAND_CODE_GET_SCHEMA_URI	0x04	GetSchemaURI
_PLDM_COMMAND_CODE_GET_RESOURCE_ETAG	0x05	GetResourceETag
_PLDM_COMMAND_CODE_RDE_OPERATION_INIT	0x10	RDEOperationInit
_PLDM_COMMAND_CODE_RDE_SUPPLY_CUSTOM_REQUEST_PARAMETERS	0x11	SupplyCustomRequestParameters
_PLDM_COMMAND_CODE_RDE_RETRIEVE_CUSTOM_RESPONSE_PARAMETERS	0x12	RetrieveCustomResponseParameters
_PLDM_COMMAND_CODE_RDE_OPERATION_COMPLETE	0x13	RDEOperationComplete
_PLDM_COMMAND_CODE_RDE_OPERATION_STATUS	0x14	RDEOperationStatus
_PLDM_COMMAND_CODE_RDE_OPERATION_KILL	0x15	RDEOperationKill
_PLDM_COMMAND_CODE_RDE_OPERATION_ENUMERATE	0x16	RDEOperationEnumerate
_PLDM_COMMAND_CODE_RDE_MULTIPART_SEND	0x30	MultipartSend
_PLDM_COMMAND_CODE_RDE_MULTIPART_RECEIVE	0x31	MultipartReceive

in.Completion_Code: Completion codes specific to the current PLDM type.

Constant	Code	Completion Code
Generic codes are included		
_PLDM_COMPLETION_CODE_RDE_ERROR_BAD_CHECKSUM	0x80	ERROR_BAD_CHECKSUM
_PLDM_COMPLETION_CODE_RDE_ERROR_CANNOT_CREATE_OPERATION	0x81	ERROR_CANNOT_CREATE_OPERATION
_PLDM_COMPLETION_CODE_RDE_ERROR_NOT_ALLOWED	0x82	ERROR_NOT_ALLOWED
_PLDM_COMPLETION_CODE_RDE_ERROR_WRONG_LOCATION_TYPE	0x83	ERROR_WRONG_LOCATION_TYPE
_PLDM_COMPLETION_CODE_RDE_ERROR_OPERATION_ABANDONED	0x84	ERROR_OPERATION_ABANDONED
_PLDM_COMPLETION_CODE_RDE_ERROR_OPERATION_UNKILLABLE	0x85	ERROR_OPERATION_UNKILLABLE
_PLDM_COMPLETION_CODE_RDE_ERROR_OPERATION_EXISTS	0x86	ERROR_OPERATION_EXISTS
_PLDM_COMPLETION_CODE_RDE_ERROR_OPERATION_FAILED	0x87	ERROR_OPERATION_FAILED
_PLDM_COMPLETION_CODE_RDE_ERROR_UNEXPECTED	0x88	ERROR_UNEXPECTED
_PLDM_COMPLETION_CODE_RDE_ERROR_UNSUPPORTED	0x89	ERROR_UNSUPPORTED
_PLDM_COMPLETION_CODE_RDE_ERROR_UNRECOGNIZED_CUSTOM_HEADER	0x90	ERROR_UNRECOGNIZED_CUSTOM_HEADER
_PLDM_COMPLETION_CODE_RDE_ERROR_ETAG_MATCH	0x91	ERROR_ETAG_MATCH

_PLDM_COMPLETION_CODE_RDE_ERROR_NO_SUCH_RESOURCE	0x92	ERROR_NO_SUCH_RESOURCE

in.Req_Schema_Class (for GetSchemaURI command)
in.Requested_Schema_Class (for GetSchemaDictionary command)

Constant	Code	schemaClass
_RDE_SCHEMA_CLASS_MAJOR	0x00	MAJOR
_RDE_SCHEMA_CLASS_EVENT	0x01	EVENT
_RDE_SCHEMA_CLASS_ANNOTATION	0x02	ANNOTATION
_RDE_SCHEMA_CLASS_COLLECTION_MEMBER_TYPE	0x03	COLLECTION_MEMBER_TYPE
_RDE_SCHEMA_CLASS_ERROR	0x04	ERROR

in.Operation_Type

Constant	Code	OperationType
_RDE_OPERATION_TYPE_OPERATION_HEAD	0x00	OPERATION_HEAD
_RDE_OPERATION_TYPE_OPERATION_READ	0x01	OPERATION_READ
_RDE_OPERATION_TYPE_OPERATION_CREATE	0x02	OPERATION_CREATE
_RDE_OPERATION_TYPE_OPERATION_DELETE	0x03	OPERATION_DELETE
_RDE_OPERATION_TYPE_OPERATION_UPDATE	0x04	OPERATION_UPDATE
_RDE_OPERATION_TYPE_OPERATION_REPLACE	0x05	OPERATION_REPLACE
_RDE_OPERATION_TYPE_OPERATION_ACTION	0x06	OPERATION_ACTION

in.Operation_status

Constant	Code	OperationStatus
_RDE_OPERATION_STATUS_OPERATION_INACTIVE	0x00	INACTIVE
_RDE_OPERATION_STATUS_OPERATION_NEEDS_INPUT	0x01	NEEDS_INPUT
_RDE_OPERATION_STATUS_OPERATION_TRIGGERED	0x02	TRIGGERED
_RDE_OPERATION_STATUS_OPERATION_RUNNING	0x03	RUNNING
_RDE_OPERATION_STATUS_OPERATION_HAVE_RESULTS	0x04	HAVE_RESULTS
_RDE_OPERATION_STATUS_OPERATION_COMPLETED	0x05	COMPLETED
_RDE_OPERATION_STATUS_OPERATION_FAILED	0x06	FAILED
_RDE_OPERATION_STATUS_OPERATION_ABANDONED	0x07	ABANDONED

in.TF

Constant	Code	TransferFlag
_RDE_TRANSFER_FLAG_START	0x00	START
_RDE_TRANSFER_FLAG_MIDDLE	0x01	MIDDLE
_RDE_TRANSFER_FLAG_END	0x02	END
_RDE_TRANSFER_FLAG_START_AND_END	0x03	START_AND_END

in.Transfer_Op

Constant	Code	TransferOperation
_RDE_XFER_FIRST_PART	0x00	XFER_FIRST_PART
_RDE_XFER_NEXT_PART	0x01	XFER_NEXT_PART
_RDE_XFER_ABORT	0x02	XFER_ABORT
_RDE_XFER_COMPLETE	0x03	XFER_COMPLETE

The maximum number of concurrent outstanding Operations the MC/Device can support for this RDE Device.
in.MC_Conc_Support:
in.Dev_Conc_Support:

MC/Device FeatureSupport flags: Operations and functionality supported by the MC/Device:
in.HS: head_supported
in.RDS: read_supported
in.CS: create_supported
in.DS: delete_supported
in.US: update_supported
in.RPLS: replace_supported
in.AS: action_supported
in.ES: events_supported

DeviceCapabilitiesFlags: Capabilities for this RDE Device; for each, 1b indicates the RDE Device has the capability, 0b not:
in.ARR: atomic_resource_read
in.ES: expand_support

in.Dev_Config_Signature: A signature (such as a CRC-32) calculated across all RDE PDRs and dictionaries that the RDE Device supports

varstirng metadata:
in.StrFmt: stringFormat { UNKNOWN = 0, ASCII = 1, UTF-8 = 2, UTF-16 = 3, UTF-16LE = 4, UTF-16BE =5 }
in.StrLen: stringLengthBytes including null terminator

in.Dev_Provider_Name: An informal name for the RDE Device

MC/Device MaximumTransferChunkSizeBytes: An indication of the maximum amount of data the MC can support for a single message transfer
in.MC_Max_Transfer_Chunk_Size_Bytes
in.Device_Max_Transfer_Chunk_Size_Bytes

in.Resource_ID: The ResourceID of any resource in the Redfish Resource PDR from which to retrieve the needed info

in.Dict_Format: The format of the dictionary as specified in the dictionary's **VersionTag**

in.Transfer_Handle: A data transfer handle that the MC shall use to retrieve the dictionary data via one or more MultipartReceive commands

in.OEM_Ext_Number: Shall be zero for a standard DMTF-published schema, or the one-based OEM extension to a standard schema

in.Str_Fragment_Count: The number of fragments N into which the URI string is broken; shall be greater than zero. The MC shall concatenate these together to reassemble the final string

in.Schema_URI: URI string fragment for the schema. The reassembled string shall be the canonical URI for the JSON Schema used by the RDE Device

in.ETAG: The ETag string data; the string text format shall be UTF-8

in.Operation_ID: Identification number for this Operation

OperationFlags Flags associated with this Operation:
in.LV: locator_valid
in.CRP: contains_request_payload
in.CCRP: contains_custom_request_parameters

in.SDTH: **SendDataTransferHandle** - handle to be used with the first MultipartSend command transferring BEJ formatted data for theoperation

in.OLL: **OperationLocatorLength -** Length in bytes of the **OperationLocator** for this Operation

in.RPL: **RequestPayloadLength** - length in bytes of the request payload in this message

in.Operation_Locator: BEJ locator indicating where the new Operation is to take place within the resource specified in **ResourceID**

in.OCP: **CompletionPercentage** 0..100: percentage complete; 101-253: reserved for future use; 254: not supported or otherwise unable to estimate (but a valid Operation) 255: invalid Operation

in.Operation_Compl_Secs: An estimate of the number of seconds remaining before the Operation is completed, or 0xFFFFFFFF if such an estimate cannot be provided

OperationExecutionFlags:
in.TS: TaskSpawned
in.HCRP: HaveCustomResponseParameters
in.HRP: HaveResultPayload
in.CA: CacheAllowed

in.RTH: **ResultTransferHandle** - a data transfer handle that the MC may use to retrieve a larger response payload via one or more **MultipartReceive** commands

PermissionFlags: Indicates the access level granted to the resource targeted by the Operation
in.Rd: read access
in.U: update access
in.Rp: replace access
in.C: create access
in.D: delete access

in.Link_Expand: The value of a $levels qualifier to a $expand query option if supplied as part of an HTTP/HTTPS GET operation

in.Col_Skip: The value of a $skip query option if supplied as part of an HTTP/HTTPS GET operation

in.Col_Top: The value of a $top query option if supplied as part of an HTTP/HTTPS GET operation

in.Pg_Offset: The page offset for paginated response data that the RDE Device supplied in conjunction with an @odata.nextlink annotation and decoded from a pagination URI

in.ETag_Op: **ETagOperation** { ETAG_IGNORE = 0; ETAG_IF_MATCH = 1; ETAG_IF_NONE_MATCH = 2 }

in.ETag_Count: Number of ETags supplied in this message

in.Hdr_Cnt: The number of custom headers being supplied in this operation

in.Header_Name: The name of the header, including the X- prefix

in.Header_Parameter: The parameter or parameters associated with the header

in.Def_Timeframe: The expected length of time in seconds before the RDE Device will be able to respond to a request to start an Operation, or 0xFF if unknown

in.New_Resource_ID: Resource ID for a newly created collection entry

KillFlags: Flags for killing the Operation
in.DR: discard_record
in.RTC: run_to_completion

in.Operation_Count: The number of active Operations N described in the remainder of this message

in.DTH: **DataTransferHandle** A handle to uniquely identify the chunk of data to be sent

in.Next_DTH: **NextDataTransferHandle** The handle for the next chunk of data for this transfer

in.Data_Len: The length in bytes N of data being sent in this chunk

in.CRC: **DataIntegrityChecksum** 32-bit CRC for the entirety of data

5.5.4 NC-SI Messages

The following values are valid for NC-SI Messages only. Undefined for other messages.

5.5.4.1 NC-SI Message Header

in.Hdr_Rev: 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.

in.IC: (MCTP integrity check bit) Indicates whether the MCTP message is covered by an overall MCTP message payload integrity check. This field is required to be the most significant bit of the first byte of the message body in the first packet of a message along with the message type bits.
0b = No MCTP message integrity check
1b = MCTP message integrity check is present

in.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.

in.Msg_Type: Defines the type of payload contained in the message data portion of the MCTP message. Is equal to _MCTP_MSG_NCSI (0x02) for each NC-SI message.

In.MCID: Management Controller ID. Should be 0x00 for DSP0222_1.1.0.

In.Payload_Length : 12 bit NC-SI payload length value.

In.Ch_Id : This 1-byte field contains the Network Controller Channel Identifier.

in.Ctrl_Packet_Type : The type of NC-SI message. Supported values: **Request**, **Response** or **AEN**.

in.Command_Code: NC-SI Command Code field identifies the type of operation the message is requesting.

Constant	Code	Command
_NC_SI_CMD_CLEAR_INITIAL_STATE	0x00	Clear Initial State
_NC_SI_CMD_SELECT_PACKAGE	0x01	Select Package
_NC_SI_CMD_DESELECT_PACKAGE	0x02	Deselect Package
_NC_SI_CMD_ENABLE_CHANNEL	0x03	Enable Channel
_NC_SI_CMD_DISABLE_CHANNEL	0x04	Disable Channel
_NC_SI_CMD_RESET_CHANNEL	0x05	Reset Channel
_NC_SI_CMD_AEN_ENABLE	0x08	AEN Enable
_NC_SI_CMD_SET_LINK	0x09	Set Link
_NC_SI_CMD_GET_LINK_STATUS	0x0A	Get Link Status
_NC_SI_CMD_GET_VERSION_ID	0x15	Get Version ID
_NC_SI_CMD_GET_CAPABILITIES	0x16	Get Capabilities
_NC_SI_CMD_GET_PARAMETERS	0x17	Get Parameters
_NC_SI_CMD_GET_CONTROLLER_PACKET_STATISTICS	0x18	Get Controller Packet Statistics
_NC_SI_CMD_GET_NC_SI_STATISTICS	0x19	Get NC-SI Statistics
_NC_SI_CMD_GET_PACKAGE_STATUS	0x1B	Get Package Status
_NC_SI_CMD_OEM_COMMAND	0x50	OEM Command
_NC_SI_CMD_GET_SUPPORTED_MEDIA	0x51	Get Supported Media
_NC_SI_CMD_CLEAR_INITIAL_STATE	0x00	Clear Initial State

In. AEN_Type:

Constant	Code	AEN Type
_NC_SI_AEN_LINK_STATUS_CHANGE	0x00	Link Status Change
_NC_SI_AEN_CONFIGURATION_REQUIRED	0x01	Configuration Required
_NC_SI_AEN_HOST_NC_DRIVER_STATUS_CHANGE	0x02	Host NC Driver Status Change
_NC_SI_AEN_MEDIUM_CHANGE	0x70	Medium change
_NC_SI_AEN_OEM_SPECIFIC_FROM	0x80	OEM specific AEN (begin of the range)
_NC_SI_AEN_OEM_SPECIFIC_TO	0xFF	OEM specific AEN (end of the range)

5.5.4.2 NC-SI Response code and reason code values

In.Response_Code : NC-SI Standard response Code.

Constant	Code	Response
_NC_SI_RESULT_CODE_COMMAND_COMPLETED	0x0000	Command Completed
_NC_SI_RESULT_CODE_COMMAND_FAILED	0x0001	Command Failed
_NC_SI_RESULT_CODE_COMMAND_UNAVAILABLE	0x0002	Command Unavailable
_NC_SI_RESULT_CODE_COMMAND_UNSUPPORTED	0x0003	Command Unsupported
_NC_SI_RESULT_CODE_COMMAND_OEM_SPECIFIC_FROM	0x8000	Vendor/OEM-specific (begin of the range)
_NC_SI_RESULT_CODE_COMMAND_OEM_SPECIFIC_TO	0xFFFF	Vendor/OEM-specific (end of the range)

In.Reason_Code: NC-SI Standard reason code.

Constant	Code	Reason
_NC_SI_REASON_CODE_NO_ERROR_NO_REASON_CODE	0x0000	No Error/No Reason Code
_NC_SI_REASON_CODE_INTERFACE_INITIALIZATION_REQUIRED	0x0001	Interface Initialization Required
_NC_SI_REASON_CODE_PARAMETER_IS_INVALID_UNSUPPORTED_OR_OUT_OFRANGE	0x0002	Parameter Is Invalid, Unsupported, or Out-of Range
_NC_SI_REASON_CODE_CHANNEL_NOT_READY	0x0003	Channel Not Ready
_NC_SI_REASON_CODE_PACKAGE_NOT_READY	0x0004	Package Not Ready
_NC_SI_REASON_CODE_INVALID_PAYLOAD_LENGTH	0x0005	Invalid payload length
_NC_SI_REASON_CODE_UNKNOWN_OR_UNSUPPORTED_COMMAND_TYPE	0x7FFF	Unknown/Unsupported Command Type
_NC_SI_REASON_CODE_OEM_REASON_CODE_FROM	0x8000	OEM Reason Code (begin of the range)
_NC_SI_REASON_CODE_OEM_REASON_CODE_TO	0xFFFF	OEM Reason Code (end of the range)

5.5.4.3 NC-SI General Message Values/Parts

In.Checksum : This 4-byte field contains the 32-bit checksum compensation value that may be included in each 1610 command and response packet by the sender of the packet.

5.5.4.4 Broadcast Packet Filter Settings

in.ARP_Pkts : ARP Packets
in.DHCP_Client_Pkts : DHCP client packets

in.DHCP_Server_Pkts: DHCP server packets

in.NetBIOS_Pkts: NetBIOS Packets

All the values are binary:

```
1 = Forward this packet type to the Management Controller.
0 = Filter out this packet type.
```

5.5.4.5 Multicast Packet Filter Settings

in.IPv6_Neighbor_Adv: IPv6 Neighbor Advertisement

in.IPv6_Router_Adv: IPv6 Router Advertisement

in.DHCPv6_Relay_And_Srv_Multicast: DHCPv6 relay and server multicast

in.DHCPv6_Multicasts_On_UDP: DHCPv6 multicasts from server to clients listening on wellknown UDP ports

in.IPv6_MLD: IPv6 MLD

in.IPv6_Neighbor_Solicitation: IPv6 Neighbor Solicitation

All the values are binary:

```
1 = Forward this packet type to the Management Controller.
0 = Filter out this packet type.
```

5.5.4.6 Link Settings

in.Auto_Negotiation: Auto Negotiation (1 = enable; 0 = disable)

in.NC_SI_10_Mbps: Link Speed (`1 = enable 10 Mbps)`

in.NC_SI_100_Mbps: Link Speed `1 = enable 100 Mbps)`

in.NC_SI_1_Gbps: Link Speed (`1 = enable 1 Gbps)`

in.NC_SI_2_5_Gbps: Link Speed (`1 = enable 2.5 Gbps)`

in.NC_SI_5_Gbps: Link Speed (`1 = enable 5 Gbps)`

in.NC_SI_10_Gbps: Link Speed (`1 = enable 10 Gbps)`

in.NC_SI_20_Gbps: Link Speed (`1 = enable 20 Gbps)`

in.NC_SI_25_Gbps: Link Speed (`1 = enable 25 Gbps)`

in.NC_SI_40_Gbps: Link Speed (`1 = enable 40 Gbps)`

in.NC_SI_50_Gbps: Link Speed (`1 = enable 50 Gbps)`

in.NC_SI_100_Gbps: Link Speed (`1 = enable 100 Gbps)`

in.Half_Duplex : `1 = enable half-duplex`

in.Full_Duplex : `1b = enable full-duplex`

in.Disable_Pause_Capability: Pause Capability (`1 = disable; 0 = enable`)

in.Asymmetric_Pause_Capability: Asymmetric Pause Capability (`1 = enable; 0 = disable`)

in.OEM_Link_Settings_Field_Valid: OEM Link Settings Field Valid (`1 = enable; 0 = disable`)

5.5.4.7 Link Status

in.Link_Flag: Link Flag (`0 = Link is down; 1 = Link is up`)

in.Speed_And_Duplex: Speed and duplex

Constant	Code	Speed And Duplex
_NC_SI_LINK_STATUS_AUTO_NEGOTIATE_NOT_COMPLETE__NO_HCD	0x0	Auto-negotiate not complete
_NC_SI_LINK_STATUS_10BASE_T_HALF_DUPLEX	0x1	10BASE-T half-duplex
_NC_SI_LINK_STATUS_10BASE_T_FULL_DUPLEX	0x2	10BASE-T full-duplex
_NC_SI_LINK_STATUS_100BASE_TX_HALF_DUPLEX	0x3	100BASE-TX half-duplex
_NC_SI_LINK_STATUS_100BASE_T4	0x4	100BASE-T4
_NC_SI_LINK_STATUS_100BASE_TX_FULL_DUPLEX	0x5	100BASE-TX full-duplex
_NC_SI_LINK_STATUS_1000BASE_T_HALF_DUPLEX	0x6	1000BASE-T half-duplex
_NC_SI_LINK_STATUS_1000BASE_T_FULL_DUPLEX	0x7	1000BASE-T full-duplex
_NC_SI_LINK_STATUS_10G_BASE_T_SUPPORT_OR_10_GBPS	0x8	10G-BASE-T support or 10 Gbps
_NC_SI_LINK_STATUS_20_GBPS	0x9	20 Gbps
_NC_SI_LINK_STATUS_25_GBPS	0xA	25 Gbps
_NC_SI_LINK_STATUS_40_GBPS	0xB	40 Gbps
_NC_SI_LINK_STATUS_50_GBPS	0xC	50 Gbps
_NC_SI_LINK_STATUS_100_GBPS	0xD	100 Gbps
_NC_SI_LINK_STATUS_2_5_GBPS	0xE	2.5 Gbps
_NC_SI_LINK_STATUS_USE_ENHANCED_SPEED_AND_DUPLEX	0xF	Use values defined in Enhanced Speed and Duplex field

in.Auto_Negotiate_Flag: Auto Negotiate Flag (`1 = Auto-negotiation is enabled`)

in.Auto_Negotiate_Complete: Auto Negotiate Complete (`1 = Auto-negotiation has completed`)

in.Parallel_Detection_Flag: Parallel Detection Flag `1b = Link partner did not support auto-negotiation and parallel detection was used to get link. This field contains 0b if Parallel Detection was not used to obtain link.`

in.LPA_1000TFD: Link Partner Advertised Speed and Duplex 1000TFD (`1 = Link Partner is 1000BASE-T full-duplex capable`)

in.LPA_1000THD:] Link Partner Advertised Speed and Duplex 1000THD (`1 = Link Partner is 1000BASE-T half-duplex capable`)

in.LPA_100T4: Link Partner Advertised Speed 100T4 (`1b = Link Partner is 100BASE-T4 capable`)

in.LPA_100TXFD: Link Partner Advertised Speed and Duplex 100TXFD (`1b = Link Partner is 100BASE-TX full-duplex capable`)

in.LPA_100TXHD: Link Partner Advertised Speed and Duplex 100TXHD (`1b = Link Partner is 100BASE-TX half-duplex capable`)

in.LPA_10TFD: Link Partner Advertised Speed and Duplex 10TFD (`1b = Link Partner is 10BASE-T full-duplex capable`)

in.LPA_10THD: Link Partner Advertised Speed and Duplex 10THD (`1b = Link Partner is 10BASE-T half-duplex capable`)

in.TX_Flow_Control: TX Flow Control Flag

```
      0 = Transmission of Pause frames by the NC onto the external network interface
is disabled.
      1 = Transmission of Pause frames by the NC onto the external network interface
is enabled.
```

in.RX_Flow_Control: RX Flow Control Flag

```
      0 = Reception of Pause frames by the NC from the external
network interface is disabled.
      1 = Reception of Pause frames by the NC from the external
network interface is enabled
```

in.LPA_Flow_Control: Link Partner Advertised Flow Control

Constant	Code	LP A Flow Control
NC_SI_LINK_STATUS_PARTNER_ADVERTISED_NOT_PAUSE_CAPABLE	0x0	Not pause capable
NC_SI_LINK_STATUS_PARTNER_ADVERTISED_SYMMETRIC_PAUSE	0x1	Symmetric pause
NC_SI_LINK_STATUS_PARTNER_ADVERTISED_ASYMMETRIC_PAUSE	0x2	Asymmetric pause
NC_SI_LINK_STATUS_PARTNER_ADVERTISED_SYMMETRIC_ASYMMETRIC_PAUSE	0x3	Symmetric/Asymmetric pause
NC_SI_LINK_STATUS_PARTNER_ADVERTISED_NOT_PAUSE_CAPABLE	0x0	Not pause capable

in.SerDes_Link:

in.OEM_Link_Speed_Valid: OEM Link Speed Valid

```
      0 = OEM link settings are invalid.
      1 = OEM link settings are valid.
```

in.Extended_Speed_And_Duplex: Extended Speed and duplex

Constant	Code	Extended Speed And Duplex
_NC_SI_LINK_STATUS_AUTO_NEGOTIATE_NOT_COMPLETE__NO_HCD	0x0	Auto-negotiate not complete
_NC_SI_LINK_STATUS_10BASE_T_HALF_DUPLEX	0x1	10BASE-T half-duplex
_NC_SI_LINK_STATUS_10BASE_T_FULL_DUPLEX	0x2	10BASE-T full-duplex
_NC_SI_LINK_STATUS_100BASE_TX_HALF_DUPLEX	0x3	100BASE-TX half-duplex
_NC_SI_LINK_STATUS_100BASE_T4	0x4	100BASE-T4
_NC_SI_LINK_STATUS_100BASE_TX_FULL_DUPLEX	0x5	100BASE-TX full-duplex

_NC_SI_LINK_STATUS_1000BASE_T_HALF_DUPLEX	0x6	1000BASE-T half-duplex
_NC_SI_LINK_STATUS_1000BASE_T_FULL_DUPLEX	0x7	1000BASE-T full-duplex
_NC_SI_LINK_STATUS_10G_BASE_T_SUPPORT_OR_10_GBPS	0x8	10G-BASE-T support or 10 Gbps
_NC_SI_LINK_STATUS_20_GBPS	0x9	20 Gbps
_NC_SI_LINK_STATUS_25_GBPS	0xA	25 Gbps
_NC_SI_LINK_STATUS_40_GBPS	0xB	40 Gbps
_NC_SI_LINK_STATUS_50_GBPS	0xC	50 Gbps
_NC_SI_LINK_STATUS_100_GBPS	0xD	100 Gbps
_NC_SI_LINK_STATUS_2_5_GBPS	0xE	2.5 Gbps
_NC_SI_LINK_STATUS_USE_ENHANCED_SPEED_AND_DUPLEX	0xF	5 Gbps

5.5.4.8 AEN Control

in.Link_Status_Change: Link Status Change AEN control (0 – disable; 1 - enable)

in.Configuration_Required: Configuration Required AEN control (0 – disable; 1 - enable)

in.Host_NC_Driver_Status_Change: Host NC Driver Status Change AEN control (0 – disable; 1 - enable)

in.OEM_Specific: OEM-specific AEN control (0 – disable; 1 - enable)

5.5.4.9 Media Descriptors

Media descriptors contain a number of Arrays for each parameter of media descriptor.

in.EID: Array of EIDs

in.Physical_Transport_Binding_Identifier: Array of Physical Transport Binding Identifier

in.Physical_Medium_Identifier: Array of Physical Medium Identifier

in.NC_SI_Pass_Through_Is_Supported: Array of values which indicate do NC-SI Pass-through is supported Each value can be 0 = is unsupported or 1 = supported.

in.Medium_Is_Available: Array of Statuses. Each value can be :
0: Medium is not available.
1: Medium is available.

in.Physical_Address_Size: Array of Physical Address Sizes

in.Physical_Address: Array of Physical Addresses

5.5.4.10 Select Package (0x01) – Request

in.HAD: Hardware arbitration.
0b = Hardware arbitration between packages is enabled.
1b = Disable hardware arbitration. Disabling hardware arbitration causes the package's arbitration logic to enter or remain in bypass mode.
In the case that the Network Controller does not support hardware arbitration, this bit is ignored.

5.5.4.11 Disable Channel command (0x04) - Request

in.ALD: 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.

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

5.5.4.12 AEN Enable command (0x08)

in.AEN_MC_ID: AEN Management controller Id

See at (**5.2.12.4.8 AEN Control**)

5.5.4.13 Set Link command (0x09)

in.OEM_Link_Settings : Vendor specified See at (**5.2.12.4.6 Link Settings**)

5.5.4.14 Set Link response (0x09)

In. Reason_Code: Command specific reason code

Constant	Code	Reason Code
_NC_SI_REASON_CODE_SET_LINK_HOST_OS__DRIVER_CONFLICT	0x0901	Set Link Host OS/ Driver Conflict
_NC_SI_REASON_CODE_SET_LINK_MEDIA_CONFLICT	0x0902	Set Link Media Conflict
_NC_SI_REASON_CODE_SET_LINK_PARAMETER_CONFLICT	0x0903	Set Link Parameter Conflict
_NC_SI_REASON_CODE_SET_LINK_POWER_MODE_CONFLICT	0x0904	Set Link Power Mode Conflict
_NC_SI_REASON_CODE_SET_LINK_SPEED_CONFLICT	0x0905	Set Link Speed Conflict
_NC_SI_REASON_CODE_SET_LINK_COMMAND_FAILED_HARDWARE_ACCESS_ERROR	0x0906	Link Command Failed-Hardware Access Error

5.5.4.15 Get Link Status response (0x8A)

See at (**5.2.12.4.7 Link Status**)

in.Host_NC_Driver_Status: Host NC Driver Status Indication

0 = The Network Controller driver for the host external network interface associated with this channel is not operational (not running), unknown, or not supported.

1 = The Network Controller driver for the host external network interface associated with this channel is being reported as operational (running).

In.OEM_Link_Status: OEM specific Link status

In. Reason_Code: Command specific reason code

Constant	Code	Reason Code
_NC_SI_REASON_CODE_GET_LINK_STATUS_FAILED_HARDWARE_ACCESS_ERROR	0x0A06	Link Command Failed-Hardware Access Error

5.5.4.16 Get Version ID Response (0x95)

in.Major : Major part of Version

in.Minor : Minor part of Version

in.Upd: Update part of Version

in.Alpha_1: Alpha 1 part of Version

in.Alpha_2: Alpha 2 part of Version

in.Firmware_Name: The name of firmware (encoded using the ISO/IEC 8859-1 Character Set)

in.Firmware_Version: The Version of Firmware

in.PCI_VID: PCI ID field

in.PCI_DID: PCI ID field

in.PCI_SVID: PCI ID field

in.PCI_SSID: PCI ID field

in.Manufacturer_ID_IANA: IANA Enterprise number

5.5.4.17 Get Capabilities response (0x96)

in.Hardware_Arbitration_Cap: Hardware Arbitration Capability (0 = 0 not supported; 1 = supported)

in.Host_NC_Driver_Status: Host NC Driver Status (0 = 0 not supported; 1 = supported)

in.NC_To_MC_Flow_Control: Network Controller to Management Controller Flow Control Support (0 = 0 not supported; 1 = supported)

in.MC_To_NC_Flow_Control : Management Controller to Network Controller Flow Control Support (0 = 0 not supported; 1 = supported)

in.All_Multicast_Addrs: All multicast addresses support

```
      0 = The channel cannot accept all multicast addresses. The channel does not
support enable/disable global multicast commands.
      1 = The channel can accept all multicast addresses. The
channel supports enable/disable global multicast
commands.
```

in.Hardware_Arbitration_Impl_Status: Hardware Arbitration Implementation Status

Constant	Code	Hardware Arbitration Implementation Status
_NC_SI_HAC_STATUS_UNKNOWN	0x0	Unknown
_NC_SI_HAC_IS_NOT_IMPLEMENTED	0x1	HAC is not implemented
_NC_SI_HAC_IS_IMPLEMENTED	0x2	HAC is implemented

See at (**5.2.12.4.4 Broadcast Packet Filter Settings**)

See at (**5.2.12.4.5 Multicast Packet Filter Settings**)

in.Buffering_Cap: Buffering Capability.

See at (**5.2.12.4.8 AEN Control**)

in.VLAN_Filter_Cnt: The number of VLAN filters, up to 15

in.Mixed_Filter_Cnt : Mixed Filter count

in.Multicast_Filter_Cnt: Multicast Filter Count

in.Unicast_Filter_Cnt : Unicast Filter count

in.VLAN_Only : 1 = VLAN shall be supported in the implementation

in.VLAN_Plus_Non_VLAN : VLAN + non-VLAN
0 = Filtering 'VLAN + non-VLAN' traffic is not supported in the implementation.
1 = Filtering 'VLAN + non-VLAN' traffic is supported in the implementation.

in.Any_VLAN_Plus_Non_VLAN : Any VLAN + non-VLAN
0 = Filtering 'Any VLAN + non-VLAN' traffic is not supported in the implementation.
1 = Filtering 'Any VLAN + non-VLAN' traffic is supported in the implementation.

in.Channel_Cnt: The number of channels supported by the Network Controller

5.5.4.18 Get Parameters response (0x97)

in.MAC_Addr_Count : The number of MAC addresses supported by the channel

in.MAC_Addr_Flags : The enable/disable state for each supported MAC address

in.VLAN_Tag_Count : The number of VLAN Tags supported by the channel

in.VLAN_Tag_Flags : The enable/disable state for each supported VLAN Tag

See at (**5.2.12.4.6 Link Settings**)

See at (**5.2.12.4.4 Broadcast Packet Filter Settings**)

in.Broadcast_Pkt_Fltr_Status : Broadcast Packet Filter status (0 = disable; 1 = enable)

in.Ch_Enabled : Channel Enabled (0 = disable; 1 = enable)

in.Ch_Net_TX_Enabled : Channel Network TX Enable (0 = disable; 1 = enable)

in.Global_Multicast_Pkt_Fltr_Status : Global Multicast Packet Filter Status (0 = disable; 1 = enable)

in.VLAN_Tag: Array of VLAN Tags

in.MAC_ADDR: Array of MAC Addresses

5.5.4.19 Get Controller Packet Statistics response (0x98)

in.Counters_Cleared_From_Last_Read_MS_Bits : Master bits for Counters Cleared from Last Read Fields

in.Counters_Cleared_From_Last_Read_LS_Bits : Least bits for Counters Cleared from Last Read Fields

in.Total_Bytes_Rcvd : Total Bytes Received

in.Total_Bytes_Tmtd: Total Bytes Transmitted

in.Total_Unicast_Pkts_Rcvd: Total Unicast Packets Received

in.Total_Multicast_Pkts_Rcvd: Total Multicast Packets Received

in.Total_Broadcast_Pkts_Rcvd: Total Broadcast Packets Received

in.Total_Unicast_Pkts_Tmtd: Total Unicast Packets Transmitted

in.Total_Multicast_Pkts_Tmtd: Total Multicast Packets Transmitted

in.Total_Broadcast_Pkts_Tmtd: Total Broadcast Packets Transmitted

in.FCS_Receive_Errs: FCS Receive Errors

in.Alignment_Errs: Alignment Errors

in.False_Carrier_Detections: False Carrier Detections

in.Runt_Pkts_Rcvd: Runt Packets Received

in.Jabber_Pkts_Rcvd: Jabber Packets Received

in.Pause_XON_Frms_Rcvd: Pause XON Frames Received

in.Pause_XOFF_Frms_Rcvd: Pause XOFF Frames Received

in.Pause_XON_Frms_Tmtd: Pause XOFF Frames Transmitted

in.Pause_XOFF_Frms_Tmtd: Pause XOFF Frames Transmitted

in.Single_Collision_Transmit_Frms: Single Collision Transmit Frames

in.Multiple_Collision_Transmit_Frms: Multiple Collision Transmit Frames

in.Late_Collision_Frms: Late Collision Frames

in.Excessive_Collision_Frms: Excessive Collision Frames

in.Control_Frms_Rcvd: Control Frames Received

in.NC_SI_64_B_Frms_Rcvd: 64 Byte Frames Received

in.NC_SI_65_127_B_Frms_Rcvd: 65–127 Byte Frames Received

in.NC_SI_128_255_B_Frms_Rcvd: 128–255 Byte Frames Received

in.NC_SI_256_511_B_Frms_Rcvd: 256–511 Byte Frames Received

in.NC_SI_512_1023_B_Frms_Rcvd: 512–1023 Byte Frames Received

in.NC_SI_1024_1522_B_Frms_Rcvd: 1024–1522 Byte Frames Received

in.NC_SI_1523_9022_B_Frms_Rcvd: 1523–9022 Byte Frames Received

in.NC_SI_64_B_Frms_Tmtd: 64 Byte Frames Transmitted

in.NC_SI_65_127_B_Frms_Tmtd: 65–127 Byte Frames Transmitted

in.NC_SI_128_255_B_Frms_Tmtd: 128–255 Byte Frames Transmitted

in.NC_SI_256_511_B_Frms_Tmtd: 256–511 Byte Frames Transmitted

in.NC_SI_512_1023_B_Frms_Tmtd: 512–1023 Byte Frames Transmitted

in.NC_SI_1024_1522_B_Frms_Tmtd: 1024–1522 Byte Frames Transmitted

in.NC_SI_1523_9022_B_Frms_Tmtd: 1523–9022 Byte Frames Transmitted

in.Valid_Bytes_Rcvd: Valid Bytes Received

in.Err_Runt_Pkts_Rcvd: Error Runt Packets Received

in.Err_Jabber_Pkts_Rcvd: Error Jabber Packets Received

5.5.4.20 Get NC-SI Statistics response (0x99)

in.NC_SI_Cmds_Rcv: NC-SI Commands Received

in.NC_SI_Ctrl_Pkts_Dropped: NC-SI Control Packets Dropped

in.NC_SI_Cmd_Type_Errs: NC-SI Unsupported Commands Received

in.NC_SI_Cmd_Checksum_Errs: NC-SI Command Checksum Errors

in.NC_SI_Rcv_Pkts: NC-SI Receive Packets

in.NC_SI_Tmt_Pkts: NC-SI Transmit Packets

in.AENs_Sent: AENs Sent

5.5.4.21 Get Package Status response (0x9B)

in.Hardware_Arbitration_Status: Hardware Arbitration Status

```
      0 = Hardware arbitration is non-operational (inactive) or unsupported.

      1 = Hardware arbitration is supported, active, and implemented for the package
on the given system.
```

5.5.4.22 OEM command (0x50)and response (0xD0)

Look at in.Manufacturer_ID_IANA described in 5.2.12.4.15 **Get Version ID Response (0x95)**

5.5.4.23 Get Supported Media response (0xD1)

in.Number_Of_Medias_Supported: Number of Medias Supported

See **(5.2.12.4.9 Media Descriptors)**

5.5.4.24 Link Status Change AEN

See (5.2.12.4.15 Get Link Status response (0x8A))

5.5.4.25 Host Network Controller Driver Status Change AEN

in.Host_NC_Driver_Status: 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).

1b = The Network Controller driver for the host external network interface associated with this channel is being reported as operational (running).

5.5.4.26 Medium change AEN

See (**5.2.12.4.23 Get Supported Media response (0xD1)**)

5.5.1 NVMe-MI Messages

The following values are valid for NVMe-MI Messages only. Undefined for other messages.
Please note that message type specific response fields are available only on MCTP Command level.

in.Data: Contains undecoded message data (for example, response specific data on Message level). Valid only if undecoded data is present.

in.SubErrors: List with specific errors associated with in.Errors list members. These lists have the same size, and *i-th* sub error entry corresponds to *i-th* in.Errors entry. See errors and their sub errors defined below.

For *_MCTP_MSG_ERROR_INVALID_LENGTH*

Constant	Code	Type
_NVME_MI_SUB_ERROR_INCOMPLETE_PAYLOAD	0x02	Incomplete Payload (less than expected)

For *_MCTP_MSG_ERROR_INVALID_FIELD_VALUE*

Constant	Code	Type
_NVME_MI_SUB_ERROR_MIC	0x01	MIC error (actual CRC32C value is different from expected)
_NVME_MI_SUB_ERROR_IC_IS_NULL	0x04	IC field is 0 (should always be set to 1 for NVMe-MI messages)

5.5.1.1 NVMe-MI Message Header

in.IC: This field is defined by the MCTP Base Specification and indicates whether the MCTP message is covered by an overall MCTP Message Integrity Check.
All NVMe-MI messages are protected by a CRC and thus this bit shall be set to '1' in all NVMe-MI messages.

in.Msg_Type: This field is defined by the MCTP Base Specification for the message type. This field shall be set to 4h in all NVMe-MI messages.

In.CSI: Command Slot Identifier: This field indicates the Command Slot with which the message is associated. For Request Messages this field indicates the Command Slot with which the Request Message is associated. For Response Messages, this field indicates the Command Slot associated with the Request Message with which the Response Message is associated.
0h - Command Slot 0
1h - Command Slot 1

in.NMIMT: NVMe-MI Message Type: This field specifies the NVMe-MI Message Type.

Constant	Code	Type
_NVME_MI_NMTMT_CONTROL_PRIMITIVE	0x00	Control Primitive
_NVME_MI_NMIMT_NVME_MI_COMMAND	0x01	NVMe-MI Command
_NVME_MI_NMIMT_NVME_ADMIN_COMMAND	0x02	NVMe Admin Command
_NVME_MI_NMIMT_PCIE_COMMAND	0x04	PCIe Command

in.ROR: Request or Response: This field indicates whether the message is a Request Message or Response Message. This field is cleared to '0' for Request Messages. This field is set to '1' for Response Messages.

Common response fields

in.Status: This field indicates the status associated with the Response Message.

Constant	Code	Type
_NVME_MI_STATUS_SUCCESS	0x00	Success
_NVME_MI_STATUS_MORE_PROCESSING_REQUIRED	0x01	More Processing Required
_NVME_MI_STATUS_INTERNAL_ERROR	0x02	Internal Error
_NVME_MI_STATUS_INVALID_COMMAND_OPCODE	0x03	Invalid Command Opcode
_NVME_MI_STATUS_INVALID_PARAMETER	0x04	Invalid Parameter
_NVME_MI_STATUS_INVALID_COMMAND_SIZE	0x05	Invalid Command Size
_NVME_MI_STATUS_INVALID_COMMAND_INPUT_DATA_SIZE	0x06	Invalid Command Input Data Size
_NVME_MI_STATUS_ACCESS_DENIED	0x07	Access Denied
_NVME_MI_STATUS_VPD_UPDATES_EXCEEDED	0x20	VPD Updates Exceeded
_NVME_MI_STATUS_PCIE_UNACCESSIBLE	0x21	PCIe Inaccessible

5.5.1.2 Control Primitives

5.5.1.2.1 Common request fields

in.OpCode: Control Primitive Opcode: This field specifies the opcode of the Control Primitive to be executed.

Constant	Code	Command
_NVME_MI_CP_OPCODE_PAUSE	0x00	Pause
_NVME_MI_CP_OPCODE_RESUME	0x01	Resume
_NVME_MI_CP_OPCODE_ABORT	0x02	Abort
_NVME_MI_CP_OPCODE_GET_STATE	0x03	Get State

_NVME_MI_CP_OPCODE_REPLAY	0x04	Replay

in.Tag: This field contains an opaque value that is passed by the Management Endpoint from the Control Primitive to the associated Response Message. The Response Message contains the same value in this field as the corresponding Request Message.

5.5.1.2.2 Get State

5.5.1.2.2.1 Request fields

in.CESF: This field specifies whether or not to clear the error state flags when completing this command.

5.5.1.2.3 Replay

5.5.1.2.3.1 Request fields

in.RRO: This field specifies the starting packet number from which the Response Message associated with the last Command Message processed in the Command Slot should be replayed.

5.5.1.3 Management Interface Command Set

5.5.1.3.1 Common request fields

in.OpCode: This field specifies the opcode of the NVMe Management Interface command to be executed

Constant	Code	Command
_NVME_MI_MI_OPCODE_READ_NVME_MI_DATA_STRUCTURE	0x00	Read NVMe-MI Data Structure
_NVME_MI_MI_OPCODE_NVM_SUBSYSTEM_HEALTH_STATUS_POLL	0x01	NVM Subsystem Health Status Poll
_NVME_MI_MI_OPCODE_CONTROLLER_HEALTH_STATUS_POLL	0x02	Controller Health Status Poll
_NVME_MI_MI_OPCODE_CONFIGURATION_SET	0x03	Configuration Set
_NVME_MI_MI_OPCODE_CONFIGURATION_GET	0x04	Configuration Get
_NVME_MI_MI_OPCODE_VPD_READ	0x05	VPD Read
_NVME_MI_MI_OPCODE_VPD_WRITE	0x06	VPD Write
_NVME_MI_MI_OPCODE_RESET	0x07	Reset

in.NMD0: This field is command specific Dword 0

in.NMD1: This field is command specific Dword 1

in.MIC: Message Integrity Check: This field contains a CRC computed over the contents of the message

5.5.1.3.2 Configuration Get

5.5.1.3.2.1 Request fields

in.CfgId: This field specifies the identifier of the Configuration that is being read.

in.PortId: This field specifies the port whose configuration is indicated.

5.5.1.3.3 Configuration Set

5.5.1.3.3.1 Request fields

5.5.1.3.3.1.1 *SMBus/I2C Frequency (Configuration Identifier 01h)*

in.CfgId: This field specifies the identifier of the Configuration that is being written.

in.SMBusFrequency: This field specifies the new frequency for the specified SMBus/I2C port.

Constant	Code	Command
_NVME_MI_SMBUS_FREQ_100_KHZ	0x01	100 kHz
_NVME_MI_SMBUS_FREQ_400_KHZ	0x02	400 kHz
_NVME_MI_SMBUS_FREQ_1_MHZ	0x03	1 MHz

in.PortId: This field specifies the port whose configuration is indicated.

5.5.1.3.3.1.2 *Health Status Change (Configuration Identifier 02h)*

in.CfgId: This field specifies the identifier of the Configuration that is being written.

in.Status_Ready: When this bit is set to '1', the corresponding bit in the Composite Controller Status field of the NVM Subsystem Health Data Structure is cleared to '0'.

in.Status_ControllerFatalStatus: When this bit is set to '1', the corresponding bit in the Composite Controller Status field of the NVM Subsystem Health Data Structure is cleared to '0'.

in.Status_ShutdownStatus: When this bit is set to '1', the corresponding bit in the Composite Controller Status field of the NVM Subsystem Health Data Structure is cleared to '0'.

in.Status_NVMSubsyatemResetOccured: When this bit is set to '1', the corresponding bit in the Composite Controller Status field of the NVM Subsystem Health Data Structure is cleared to '0'.

in.Status_ControllerEnableChangeOccured: When this bit is set to '1', the corresponding bit in the Composite Controller Status field of the NVM Subsystem Health Data Structure is cleared to '0'

in.Status_NamespaceAttributeChanged: When this bit is set to '1', the corresponding bit in the Composite Controller Status field of the NVM Subsystem Health Data Structure is cleared to '0'

in.Status_FirmwareActivated: When this bit is set to '1', the corresponding bit in the Composite Controller Status field of the NVM Subsystem Health Data Structure is cleared to '0'.

in.Status_ControllerStatusChange: When this bit is set to '1', the corresponding bit in the Composite Controller Status field of the NVM Subsystem Health Data Structure is cleared to '0'.

in.Status_CompositeTemperatureChange: When this bit is set to '1', the corresponding bit in the Composite Controller Status field of the NVM Subsystem Health Data Structure is cleared to '0'.

in.Status_PercentageUsed: When this bit is set to '1', the corresponding bit in the Composite Controller Status field of the NVM Subsystem Health Data Structure is cleared to '0'.

in.Status_AvailableSpare: When this bit is set to '1', the corresponding bit in the Composite Controller Status field of the NVM Subsystem Health Data Structure is cleared to '0'

in.Status_CriticalWarning: When this bit is set to '1', the corresponding bit in the Composite Controller Status field of the NVM Subsystem Health Data Structure is cleared to '0'.

5.5.1.3.3.1.3 *MCTP Transmission Unit Size (Configuration Identifier 03h)*

in.CfgId: This field specifies the identifier of the Configuration that is being written

in.PortId: This field specifies the port whose MCTP Transmission Unit Size is specified.

in. MctpTransmissionUnitSize: This field contains the MCTP Transmission Unit Size in bytes to be used by the port.

5.5.1.3.4 Controller Health Status Poll

5.5.1.3.4.1 Request fields

in.CTLID: This field specifies the starting Controller ID from which to return health status information.

in.MAXRENT: This field specifies the maximum number of Controller Health Data Structure entries that may be returned in the completion. This is 0's based field. The maximum number of entries is 255. Specifying 256 entries is interpreted as an Invalid Field.

in.INCF: When this bit is set to 1, Controller Health Status is reported for NVMe Controllers associated with a non SR-IOV PCI Function.

in.INCPF: When this bit is set to 1, Controller Health Status is reported for NVMe Controllers associated with SR-IOV Physical Functions (PFs)

in.INCVF: When this bit is set to 1, Controller Health Status is reported for NVMe Controllers associated with SR-IOV Virtual Functions (VFs)

in.ALL: When this bit is set to '1', health status is returned for Controllers regardless of the status of the Health Status Changed flag bit vector (i.e., it is as though all the bits are set in the Health Status Changed flag bit vector).

in.CSTS: When this bit is set to 1, Controller status changes are reported.

in.CTEMP: When this bit is set to 1, composite temperature changes are reported.

in.PDLU: When this bit is set to 1, percentage used changes are reported.

in.SPARE: When this bit is set to 1, available spare changes are reported.

in.CWARN: When this bit is set to 1, critical warning changes are reported.

in.CCF: When this bit is set to 1, the state of reported changed flag bits in the changed flag bit vector are cleared in Controllers whose health status is contained in the Response Data.

5.5.1.3.5 NVM Subsystem Health Status Poll

5.5.1.3.5.1 Request fields

in.CS: When this bit is set to 1, the state of reported Composite Controller Status is cleared.

5.5.1.3.6 Read NVMe-MI Data Structure

5.5.1.3.6.1 Request fields

in.CTRLID: This field contains the Controller identifier whose data structure is returned.

in.PORTID: This field contains the identifier of the port whose data structure is returned.

in.DTYP: This field specifies the data structure to return.

Constant	Code	Command
_NVME_MI_DTYP_NVM_SUBSYSTEM_INFO	0x00	NVM Subsystem Information
_NVME_MI_DTYP_PORT_INFO	0x01	Port Information
_NVME_MI_DTYP_CONTROLLER_LIST	0x02	Controller List
_NVME_MI_DTYP_CONTROLLER_INFO	0x03	Controller Information
_NVME_MI_DTYP_OPTIONAL_COMMANDS_SUPPORTED	0x04	Optional Commands Supported

5.5.1.3.7 Reset

5.5.1.3.7.1 Request fields

in.ResetType: This field specifies the type of reset to be performed.

Constant	Code	Command
_NVME_MI_RESET_TYPE_RESET_NVM_SUBSYSTEM	0x00	Reset NVM Subsystem

5.5.1.3.8 VPD Read

5.5.1.3.8.1 Request fields

in.DOFST: This field specifies the starting offset, in bytes, into the VPD data that is contained in the Response Message.

in.DLEN: This field specifies the length, in bytes, to be read from the VPD starting at the byte offset specified by DOFST.

5.5.1.3.9 VPD Write

5.5.1.3.9.1 Request fields

in.DOFST: This field specifies the starting offset, in bytes, into the VPD data that is written.

in.DLEN: This field specifies the length, in bytes, to be written to the VPD starting at the byte offset specified by DOFST.

in.VPD: Vital Product Data written.

5.5.1.4 NVM Express Admin Command Set

5.5.1.4.1 Common request fields

in. OpCode: This field specifies the opcode of the command to be executed. Refer to the NVMe specification.

Constant	Code	Command
_NVME_MI_ADM_OPCODE_FIRMWARE_ACTIVATE	0x10	Firmware Activate/Commit
_NVME_MI_ADM_OPCODE_FIRMWARE_IMAGE_DOWNLOAD	0x11	Firmware Image Download
_NVME_MI_ADM_OPCODE_FORMAT_NVM	0x80	Format NVM
_NVME_MI_ADM_OPCODE_GET_FEATURES	0x0A	Get Features
_NVME_MI_ADM_OPCODE_GET_LOG_PAGE	0x02	Get Log Page
_NVME_MI_ADM_OPCODE_IDENTIFY	0x06	Identify
_NVME_MI_ADM_OPCODE_NAMESPACE_MANAGEMENT	0x0D	Namespace Management
_NVME_MI_ADM_OPCODE_NAMESPACE_ATTACHMENT	0x15	Namespace Attachment
_NVME_MI_ADM_OPCODE_SECURITY_SEND	0x81	Security Send
_NVME_MI_ADM_OPCODE_SECURITY_RECEIVE	0x82	Security Receive
_NVME_MI_ADM_OPCODE_SET_FEATURES	0x09	Set Features

in. NSID: This field specifies the namespace ID that this command applies to.

in. MPTR: This field contains the address of a contiguous physical buffer of metadata.

in.DLEN_CFLG: if set to '1' then the command contains a length value in bytes 32-35. If cleared to '0' then the DLEN field shall be cleared to 0h.

in.DOFST_CFLG: if set to '1' then the command contains an offset value in bytes 28-31. If cleared to '0' then the DOFST field shall be cleared to 0h.

in.DOFST: For commands that transmit data from the Management Controller to the Management Endpoint or do not transmit data, this field shall be cleared to '0'. For commands that transmit data from the Management Endpoint to the Management Controller, this field specifies the starting offset, in bytes, into the completion data contained in the Response Message.

in.DLEN: For commands that do not transmit data in neither the Request Message nor Response Message, this field shall be cleared to 0h. For commands that transmit data from the Management Controller to the Management Endpoint, this field specifies the length, in bytes, of the data contained in the Request Message.

in.PRP2: This field can be reserved, single page base address or PRP list pointer, depending on the command and data transfer size.

5.5.1.4.2 Get Features

5.5.1.4.2.1 Request fields

in.SEL: This field specifies which value of the attributes to return in the provided data.

Constant	Code	Command
_NVME_MI_SEL_CURRENT	0x00	Current
_NVME_MI_SEL_DEFAULT	0x01	Default
_NVME_MI_SEL_SAVED	0x02	Saved
_NVME_MI_SEL_SUPPORTED_CAPABILITIES	0x03	Supported Capabilities

in.FID: This field specifies the identifier of the Feature for which to provide data.

Constant	Code	Command
_NVME_MI_FID_ARBITRATION	0x01	Arbitration
_NVME_MI_FID_POWER_MANAGEMENT	0x02	Power Management
_NVME_MI_FID_LBA_RANGE_TYPE	0x03	LBA Range Type
_NVME_MI_FID_TEMPERATURE_THRESHOLD	0x04	Temperature Threshold
_NVME_MI_FID_ERROR_RECOVERY	0x05	Error Recovery

_NVME_MI_FID_VOLATILE_WRITE_CACHE	0x06	Volatile Write Cache
_NVME_MI_FID_NUMBER_OF_QUEUES	0x07	Number of Queues
_NVME_MI_FID_INTERRUPT_COALESCING	0x08	Interrupt Coalescing
_NVME_MI_FID_INTERRUPT_VECTOR_CONFIGURATION	0x09	Interrupt Vector Configuration
_NVME_MI_FID_WRITE_ATOMICITY	0x0A	Write Atomicity
_NVME_MI_FID_ASYNC_EVENT_CONFIGURATION	0x0B	Asynchronous Event Configuration
_NVME_MI_FID_AUTOMONOUS_POWER_STATE_TRANSITION	0x0C	Autonomous Power State Transition
_NVME_MI_FID_HOST_MEMORY_BUFFER	0x0D	Host Memory Buffer
_NVME_MI_FID_SOFTWARE_PROGRESS_MARKER	0x80	Software Progress Marker
_NVME_MI_FID_HOST_IDENTIFIER	0x81	Host Identifier
_NVME_MI_FID_RESERV_NOTIFICATION_MASK	0x82	Reservation Notification Mask
_NVME_MI_FID_RESERV_PERSISTANCE	0x83	Reservation Persistence

5.5.1.4.3 Get Log Page

5.5.1.4.3.1 Request fields

in.NUMD: This field specifies the number of Dwords to return.

in.LID: This field specifies the identifier of the log page to retrieve.

Constant	Code	Command
_NVME_MI_LID_ERROR_INFO	0x01	Error Information
_NVME_MI_LID_SMART_HEALTH_INFO	0x02	SMART / Health Information
_NVME_MI_LID_FIRMWARE_SLOT_INFO	0x03	Firmware Slot Information
_NVME_MI_LID_CHANGED_NAMESPACE_LIST	0x04	Changed Namespace List
_NVME_MI_LID_COMMAND_EFFECTS_LOG	0x05	Command Effects Log

5.5.1.4.4 Identify

5.5.1.4.4.1 Request Fields

in.CNTID: This field specifies the number of Dwords to return.

in.CNS: This field specifies the identifier of the log page to retrieve.

Constant	Code	Command
_NVME_MI_CNS_NAMESPACE_ATTACHED	0x00	Namespace (attached)
_NVME_MI_CNS_CONTROLLER	0x01	Controller
_NVME_MI_CNS_NAMESPACE_LIST_ATTACHED	0x02	Namespace List (attached)
_NVME_MI_CNS_NAMESPACE_LIST_ALL	0x10	Namespace List (all)
_NVME_MI_CNS_NAMESPACE_ALL	0x11	Namespace (all)
_NVME_MI_CNS_CONTROLLER_LIST_ATTACHED	0x12	Controller List (attached)
_NVME_MI_CNS_CONTROLLER_LIST_ALL	0x13	Controller List (all)

5.6 MCTP Commands specific set of members

Valid for MCTP commands only. Undefined for other levels.

MCTP Command level allows to select MCTP Message sub-transaction by calling SelectSubResponse (num) and SelectSubRequest (num). Specific request or response fields can be accessed through “Request_” or

“Response_” prefix. For list of available fields see MCTP Message specific set of members and specific command response members listed below.

For example, following code allows to obtain EID pool size for the 3rd response:

```
if (in.SubResponsesAmount > 2)
{
	SelectSubResponse(2);
	eid_pool_size = in.Response_EID_Pool_Size;
}
```

SelectSubRequest (num): selects MCTP Message request #num inside current MCTP Command transaction. Zero based value. Should be less than in.SubRequestsAmount. By default the first request is selected. Returns 1 while selection was successful

SelectSubResponse (num): selects MCTP Message response #num inside current MCTP Command transaction. Zero based value. Should be less than in.SubResponsesAmount. By default the first response is selected. Returns 1 while selection was successful

in.ChId: Channel ID field. Applicable to NC-SI only.

in.CommandCode: Command code.

in.CompletionCode: Completion code. Depends on currently response selected. See in.ResponseNum .

in.CSI: CSI field. Applicable to NVMe-MI only.

in.D: D field. Applicable to PLDM only.

in.DstAddr: destination’s physical address. Depends on transport binding.

in.DstEndpointId: destination’s Endpoint Id.

in.IId: IID field. Applicable to NC-SI only.

in.InstanceId: Instance ID field. Applicable to MCTP Ctrl and PLDM only.

in.IsChainedTransfer: If current MCTP Cmd contains messages chain (table transaction). Returns 0 or 1. Applicable to PLDM only.

in.IsMctpCmdDeviceToHost: command’s direction is from device to host. Applicable to PCIe VDM transport binding only. Returns 0 or 1.

in.IsMctpCmdError: if current MCTP Cmd transaction contains error. Returns 0 or 1.

in.IsMctpCmdHostToDevice: command’s direction is from host to device. Applicable to PCIe VDM transport binding only. Returns 0 or 1.

in.IsMctpCtrl: if current MCTP message type is MCTP Control. Returns 0 or 1.

in.IsNCSI: if current MCTP message type is NC-SI over MCTP. Returns 0 or 1.

in.IsNVMeMI: if current MCTP message type is NVM Express Management Messages over MCTP. Returns 0 or 1.

in.IsPCIeVDM: if current transport binding is PCIe VDM. Returns 0 or 1.

in.IsPLDM: if current MCTP message type is Platform Level Data Model (PLDM). Returns 0 or 1.

in.IsSMBus: if current transport binding is SMBus. Returns 0 or 1.

in.MctpCmdDirection: command's direction:

Constant	Code	Type
_MCTP_CMD_DIR_H2D	0x00	Downstream (host to device)
_MCTP_CMD_DIR_D2H	0x01	Upstream (device to host)
_MCTP_CMD_DIR_SMBUS	0x02	SMBus

in.MctpCmdType: MCTP Command type:

Constant	Code	Type
_MCTP_CMD_TYPE_INCOMPLETE_REQUEST	0x00	Error : Incomplete Request
_MCTP_CMD_TYPE_INCOMPLETE_RESPONSE	0x01	Error : Incomplete Response
_MCTP_CMD_TYPE_INCOMPLETE_CHAIN	0x02	Error : Incomplete Chain (table transaction)
_MCTP_CMD_TYPE_REQUEST_RESPONSE	0x03	Request - Response
_MCTP_CMD_TYPE_BROADCAST	0x04	Broadcast
_MCTP_CMD_TYPE_BROADCAST_DATAGRAM	0x05	Broadcast Datagram
_MCTP_CMD_TYPE_CHAIN	0x06	Chain (table transaction)
_MCTP_CMD_TYPE_INCOMPLETE_START_TABLE	0x07	Error : Incomplete Table, no start found
_MCTP_CMD_TYPE_INCOMPLETE_END_TABLE	0x08	Error : Incomplete Table, no end found
_MCTP_CMD_TYPE_MATCH_ERROR_EID	0x09	Error : EID are not matched

in. MctpMsgRequestIndex: current MCTP Message request index. MCTP Message request should be previously selected by calling SelectSubRequest function. If not, first request will be used.

in. MctpMsgResponseIndex: current MCTP Message response index. MCTP Message response should be previously selected by calling SelectSubResponse function. If not, first response will be used.

in.MctpMsgType: MCTP message type:

Constant	Code	Type
_MCTP_MSG_CONTROL	0x00	MCTP Control
_MCTP_MSG_PLDM	0x01	Platform Level Data Model (PLDM)
_MCTP_MSG_NC_SI_OVER_MCTP	0x02	NC-SI over MCTP
_MCTP_MSG_NVME_OVER_MCTP	0x04	NVM Express Management Messages over MCTP (NVMe-MI)

in.NMIMT: NMIMT field. Applicable to NVMe-MI only.

in.PLDMType: PLDM message type. Applicable to PLDM only.

in.ReasonCode: Reason code. Applicable to NC-SI only.

in.ResponseCode: Response code. Applicable to NC-SI and NVMe-MI only.

in.Rq: Rq field. Applicable to PLDM only.

in.SrcAddr: source's physical address. Depends on transport binding.

in.SrcEndpointId: source's Endpoint Id.

in.SubRequestsAmount: amount of MCTP Message requests in current command. Useful when calling SelectSubRequest.

in.SubResponsesAmount: amount of MCTP Message responses in current command. Useful when calling SelectSubResponse.

in.Transport: current transport binding:

Constant	Code	Type
_MCTP_CMD_TRANSPORT_PCIE_VDM	0x02	MCTP over PCIe VDM
_MCTP_CMD_TRANSPORT_SMBUS	0x01	MCTP over SMBus

5.6.1.1 NVMe-MI specific sub-transaction response fields

Due to NVMe-MI protocol design command-specific NVMe-MI response fields are not available on MCTP Message, but only on MCTP Command level. These fields shall be accessed using *in.Response_<FieldName>* notation as explained above. Array members should be accessed using zero-based indexes in square brackets, like this: *in.ArrayField[0]*.

5.6.1.1.1 Control Primitives

5.6.1.1.1.1 Pause

in.Tag: This field contains an opaque value that is sent from the Management Controller in the Control Primitive Request Message and returned by the Management Endpoint in to the associated response.

in.PFSS0: This field indicates whether or not Command Slot 0 is paused after completing the Pause primitive. A '1' in this field indicates the Command Slot is paused. A '0' in this field indicates the Command Slot is not paused.

in.PFSS1: This field indicates whether or not Command Slot 1 is paused after completing the Pause primitive. A '1' in this field indicates the Command Slot is paused. A '0' in this field indicates the Command Slot is not paused.

5.6.1.1.1.2 Abort

in.Tag: This field contains an opaque value that is sent from the Management Controller in the Control Primitive Request Message and returned by the Management Endpoint in to the associated response.

in.CPAS: This field indicates the effect of the Abort primitive on the processing of the Command Message associated with the Command Slot.

5.6.1.1.1.3 Get State

in.Tag: This field contains an opaque value that is sent from the Management Controller in the Control Primitive Request Message and returned by the Management Endpoint in to the associated response.

in.SSTA: This field indicates the current state of the Command Slot.

Constant	Code	Command
_NVME_MI_CONTROL_PRIMITIVES_SSTA_IDLE	0x00	Idle
_NVME_MI_CONTROL_PRIMITIVES_SSTA_RECEIVE	0x01	Receive
_NVME_MI_CONTROL_PRIMITIVES_SSTA_PROCESS	0x02	Process
_NVME_MI_CONTROL_PRIMITIVES_SSTA_TRANSMIT	0x03	Transmit

in.CMNICS: This field is set to '1' if the Management Endpoint received a Command Message packet while the Command Slot is not in the Idle state since the last time Get State primitive was executed with the CESF field set to '1'.

in.TMICE: This field is set to '1' if the Management Endpoint detected an error of this type (refer to the MCTP Base Specification) since the last time Get State primitive was executed with the CESF field set to '1'.

in.WPTT: This field is set to '1' if the Management Endpoint detected an error of this type (refer to the MCTP Base Specification) since the last time Get State primitive was executed with the CESF field set to '1'.

in.UTUNT: This field is set to '1' if the Management Endpoint detected an error of this type (refer to the MCTP Base Specification) since the last time Get State primitive was executed with the CESF field set to '1'.

in.BHVS: This field is set to '1' if the Management Endpoint detected an error of this type (refer to the MCTP Base Specification) since the last time Get State primitive was executed with the CESF field set to '1'.

in.UDSTID: This field is set to '1' if the Management Endpoint detected an error of this type (refer to the MCTP Base Specification) since the last time Get State primitive was executed with the CESF field set to '1'.

in.ITU: This field is set to '1' if the Management Endpoint detected an error of this type (refer to the MCTP Base Specification) since the last time Get State primitive was executed with the CESF field set to '1'.

in.UMEP: This field is set to '1' if the Management Endpoint detected an error of this type (refer to the MCTP Base Specification) since the last time Get State primitive was executed with the CESF field set to '1'.

in.OSPSN: This field is set to '1' if the Management Endpoint detected an error of this type (refer to the MCTP Base Specification) since the last time Get State primitive was executed with the CESF field set to '1'

in.BUEMT: This field is set to '1' if the Management Endpoint detected an error of this type (refer to the MCTP Base Specification) since the last time Get State primitive was executed with the CESF field set to '1'.

in.BPOPL: This field is set to '1' if a packet sent to the Management Endpoint failed a transport specific packet integrity check since the last time Get State primitive was executed with the CESF field set to '1'.

in.NSSRO: This field indicates when an NVM Subsystem Reset occurs while main power is applied. This field is set to '1' if the last occurrence of an NVM Subsystem Reset occurred while main power was applied to the NVM Subsystem. This field is cleared to '0' following a power cycle and following a Get State primitive with the CESF field set to '1'

in.PFLG: This field indicates whether or not the Command Slot is paused. A '1' in this field indicates the Command Slot is paused. A '0' in this field indicates the Command Slot is not paused.

5.6.1.1.1.4 Replay

in.Tag: This field contains an opaque value that is sent from the Management Controller in the Control Primitive Request Message and returned by the Management Endpoint in to the associated response.

in.RR: This bit indicates if a previous Response Message is retransmitted.

5.6.1.1.2 Management Interface Command Set

5.6.1.1.2.1 Configuration Get

5.6.1.1.2.1.1 SMBus/I2C Frequency (Configuration Identifier 01h)

in.SMBusFrequency: The current frequency of the SMBus/I2C. The default value for this field following a reset or power cycle is 1h, if SMBus is supported.

Constant	Code	Command
_NVME_MI_SMBUS_FREQ_NOT_SUPPORTED_DISABLED	0x00	SMBus is not supported or is disabled
_NVME_MI_SMBUS_FREQ_100_KHZ	0x01	100 kHz
_NVME_MI_SMBUS_FREQ_400_KHZ	0x02	400 kHz
_NVME_MI_SMBUS_FREQ_1_MHZ	0x03	1 MHz

5.6.1.1.2.1.2 MCTP Transmission Unit Size (Configuration Identifier 03h)

in. MctpTransmissionUnitSize: This field contains the MCTP Transmission Unit Size in bytes to be used by the port. The default value for this field following a reset or power cycle is 40h (64).

5.6.1.1.2.2 Controller Health Status Poll

in.RENT: This field specifies the number of Controller Health Data Structure
Entries present in the Response Data for this Response Message. This is a 0's based value.

in.CTLID: This field specifies the NVMe Controller identifier with which the data contained in this data structure is associated.

in.Status_Ready: This bit corresponds to the value of the Ready (CSTS.RDY) bit

in.Status_ControllerFatalStatus: This bit corresponds to the value of the Controller Fatal Status (CSTS.CFS) bit.

in.Status_ShutdownStatus: This field corresponds to the value of the Shutdown Status (CSTS.SHST) field.

in.Status_NVMSubsyatemResetOccured: This bit corresponds to the value of the NVM Subsystem Reset Occurred (CSTS.NSSRO) bit.

in.Status_ControllerEnableChangeOccured: This bit is set to '1' when the Enable (CC.EN) bit changes state. This bit is cleared to '0' after it is read using this command.

in.Status_NamespaceAttributeChanged: This bit is set to '1' when a change occurs in the Identify Namespace data structure for one or more namespaces. This bit is cleared to '0' after it is read using this command.

in.Status_FirmwareActivated: This bit is set to '1' when a new firmware image is activated. This bit is cleared to '0' after it is read using this command.

in.CTEMP: This field contains a value corresponding to a temperature in degrees Kelvin that represents the current composite temperature of the Controller and namespace(s) associated with that Controller. The value of this field corresponds to the value in the NVMe Controller SMART / Health Information Log

in.PDLU: This field contains a vendor specific estimate of the percentage of NVM Subsystem life used based on the actual usage and the manufacturer's prediction of NVM life. The value of this field corresponds to the value in the NVMe Controller SMART / Health Information Log.

in.SPARE: This field contains a normalized percentage (0 to 100%) of the remaining spare capacity available. The value of this field corresponds to the value in the NVMe Controller SMART / Health Information Log

in.Status_SpareThreshold: This bit is set to '1' when the available spare has fallen below the available spare threshold

in.Status_AboveOrUnderThreshold: This bit is set to '1' when a temperature is above an over temperature threshold or below an under temperature threshold.

in.Status_ReliabilityDegraded: This bit is set to '1' when NVM Subsystem reliability has been degraded due to significant media related errors or an internal error.

in.Status_ReadOnly: This bit is set to '1' when the media has been placed in read only mode

in.Status_VolatileMemoryBackupFailed: This bit is set to '1' when the volatile memory backup device has failed.

5.6.1.1.2.3 NVM Subsystem Health Status Poll

in.Port1PCIeLinkActive: This bit is set to '1' to indicate the second port's PCIe link is up. If cleared to '0', then the second port's PCIe link is down or not present..

in.Port0PCIeLinkActive: This bit is set to '1' to indicate the first port's PCIe link is up (i.e., the Data Link Control and Management State Machine is in the DL_Active state). If cleared to '0', then the PCIe link is down.

in.ResetNotRequired: This bit is set to '1' to indicate the NVM Subsystem does not need a reset to resume normal operation. If cleared to '0' then the NVM Subsystem has experienced an error that prevents continued normal operation. A Controller Level Reset is required to resume normal operation.

in.DriveFunctional: This bit is set to '1' to indicate an NVM Subsystem is functional. If cleared to '0', then there is an unrecoverable failure in the NVM Subsystem and the rest of the transmission may contain invalid information.

in.SmartWarnings: This field contains the Critical Warning field (byte 0) of the NVMe SMART / Health Information log. Each bit in this field is inverted from the NVMe definition (i.e., the management interface shall indicate a '0' value while the corresponding bit is '1' in the log page).

in.CompositeTemperature: This field indicates the current temperature in degrees Celsius. If a temperature value is reported, it should be the same temperature as the Composite Temperature from the SMART log of hottest Controller in the NVM Subsystem. The reported temperature range is vendor specific.

Constant	Code	Command
_NVME_MI_COMPOSITE_TEMPERATURE_127C_OR_HIGHER	0x7F	127C or higher
_NVME_MI_COMPOSITE_TEMPERATURE_NO_OR_OLD_DATA	0x80	No temperature data or temperature data is more the 5 seconds old
_NVME_MI_COMPOSITE_TEMPERATURE_SENSOR_FAILURE	0x81	Temperature sensor failure
_NVME_MI_COMPOSITE_TEMPERATURE_MINUS_60C_OR_LOWER	0xC4	Temperature is -60C or lower

in.PercentageDriveLifeUsed: Contains a vendor specific estimate of the percentage of NVM Subsystem NVM life used based on the actual usage and the manufacturer's prediction of NVM life.

in.Status_Ready: This bit is set to '1' when the value of the Ready (CSTS.RDY) bit transitions from a '0' to a '1' in one or more Controllers in the NVM Subsystem.

in.Status_ControllerFatalStatus: This bit is set to '1' when the value of the Controller Fatal Status (CSTS.CFS) bit transitions from a '0' to a '1' in one or more Controllers in the NVM Subsystem.

in.Status_ShutdownStatus: This bit is set to '1' when the value of the Shutdown Status (CSTS.SHST) field bit transitions from a '0' to a '1' in one or more Controllers in the NVM Subsystem.

in.Status_NVMSubsyatemResetOccured: This bit is set to '1' when the value of the NVM Subsystem Reset Occurred (CSTS.NSSRO) bit transitions from a '0' to a '1' in one or more Controllers in the NVM Subsystem.

in.Status_ControllerEnableChangeOccured: This bit is set to '1' when the Enable (CC.EN) bit changes state (i.e., '0' to '1' or '1' to '0') in one or more Controllers in the NVM Subsystem.

in.Status_NamespaceAttributeChanged: This bit is set to '1' when a change occurred in the Identify Namespace data structure associated with one or more namespaces in the NVM Subsystem.

in.Status_FirmwareActivated: This bit is set to '1' when a new firmware image is activated in the NVM Subsystem.

in.Status_ControllerStatusChange: This bit is set to '1' when the Controller Status field in the Controller Health Data Structure is set to '1' in one or more Controllers in the NVM Subsystem.

in.Status_CompositeTemperatureChange: This bit is set to '1' when the Composite Temperature field in the Controller Health Data Structure is set to '1' in one or more Controllers in the NVM Subsystem.

in.Status_PercentageUsed: This bit is set to '1' when the Percentage Used field in the Controller Health Data Structure is set to '1' in one or more Controllers in the NVM Subsystem.

in.Status_AvailableSpare: This bit is set to '1' when the Available Spare field in the Controller Health Data Structure has changed state in one or more Controllers in the NVM Subsystem.

in.Status_CriticalWarning: This bit is set to '1' when the Critical Warning field in the Controller Health Data Structure is set to '1' in one or more Controllers in the NVM Subsystem.

5.6.1.1.2.4 Read NVMe-MI Data Structure

in. ResponseDataLength: The length, in bytes, of the Response Data field in this Response Message.

5.6.1.1.2.4.1 NVM Subsystem Information Data Structure

in.NUMP: This field specifies the maximum number of ports of any type supported by the NVM Subsystem. This is a 0's based value.

in.MJR: This field shall be set to 1h to indicate the major version number of this specification.

in.MNR: This field shall be cleared to 0h to indicate the minor version number of this specification.

5.6.1.1.2.4.2 Port Information Data Structure

in.PortType: Specifies the port type.

Constant	Code	Command
_NVME_MI_PORT_TYPE_INACTIVE	0x00	Inactive
_NVME_MI_PORT_TYPE_PCIE	0x01	PCIe
_NVME_MI_PORT_TYPE_SMBUS	0x02	SMBus

in.MaxMctpTransmissionUnitSize: The maximum MCTP Transmission Unit size the port is capable of sending and receiving.

5.6.1.1.2.4.2.1 PCIe Port Specific Data

in.PCIeMaxPayloadSize: This field indicates the Max Payload Size for the specified PCIe port. If the link is not active, this field should be cleared to 0h.

Constant	Code	Command
_NVME_MI_PCIE_MAX_PAYLOAD_SIZE_128B	0x00	128 bytes
_NVME_MI_PCIE_MAX_PAYLOAD_SIZE_256B	0x01	256 bytes
_NVME_MI_PCIE_MAX_PAYLOAD_SIZE_512B	0x02	512 bytes
_NVME_MI_PCIE_MAX_PAYLOAD_SIZE_1024B	0x03	1024 bytes
_NVME_MI_PCIE_MAX_PAYLOAD_SIZE_2048B	0x04	2048 bytes
_NVME_MI_PCIE_MAX_PAYLOAD_SIZE_4096B	0x05	4096 bytes

in.PCIeSupportedLinkSpeeds: This field indicates the Supported Link Speeds for the specified PCIe port.

in.PCIeCurrentLinkSpeed: The port's PCIe negotiated link speed using the same encoding as the PCIe Supported Link Speed Vector field. A value of 0h in this field indicates the PCIe Link is not available

in.PCIeMaxLinkWidth: The maximum PCIe link width for this NVM Subsystem port. This is the expected negotiated link width that the port link trains to if the platform supports it.

Constant	Code	Command
_NVME_MI_PCIE_LINK_WIDTH_X1	1	PCIe x1
_NVME_MI_PCIE_LINK_WIDTH_X2	2	PCIe x2
_NVME_MI_PCIE_LINK_WIDTH_X4	4	PCIe x4
_NVME_MI_PCIE_LINK_WIDTH_X8	8	PCIe x8
_NVME_MI_PCIE_LINK_WIDTH_X12	12	PCIe x12
_NVME_MI_PCIE_LINK_WIDTH_X16	16	PCIe x16
_NVME_MI_PCIE_LINK_WIDTH_X32	32	PCIe x32

in.PCIeNegotiatedLinkWidth: The negotiated PCIe link width for this port.

Constant	Code	Command
_NVME_MI_PCIE_LINK_WIDTH_NOT_ACTIVE	0	Link not active
_NVME_MI_PCIE_LINK_WIDTH_X1	1	PCIe x1
_NVME_MI_PCIE_LINK_WIDTH_X2	2	PCIe x2
_NVME_MI_PCIE_LINK_WIDTH_X4	4	PCIe x4
_NVME_MI_PCIE_LINK_WIDTH_X8	8	PCIe x8
_NVME_MI_PCIE_LINK_WIDTH_X12	12	PCIe x12
_NVME_MI_PCIE_LINK_WIDTH_X16	16	PCIe x16
_NVME_MI_PCIE_LINK_WIDTH_X32	32	PCIe x32

5.6.1.1.2.4.2.2 SMBus Port Specific Data

in.CurrentVPDSmbusAddress: This field indicates the current VPD SMBus/I2C address. A value of 0h indicates there is no VPD.

in.MaxVPDAccessSmbusFrequency: This field indicates the maximum SMBus/I2C frequency supported on the VPD interface.

Constant	Code	Command
_NVME_MI_VPD_MAX_SMBUS_FREQ_NOT_SUPPORTED	0x00	Not supported
_NVME_MI_VPD_MAX_SMBUS_FREQ_100_KHZ	0x01	100 kHz
_NVME_MI_VPD_MAX_SMBUS_FREQ_400_KHZ	0x02	400 kHz
_NVME_MI_VPD_MAX_SMBUS_FREQ_1_MHZ	0x03	1 MHz

in.CurrentManagementEndpointSmbusAddress: This field indicates the current MCTP SMBus/I2C address. A value of 0h indicates there is no Management Endpoint on this port.

in.MaxManagementEndpointSmbusFrequency: This field indicates the maximum SMBus/I2C frequency supported by the Management Endpoint.

Constant	Code	Command
_NVME_MI_VPD_MAX_SMBUS_FREQ_NOT_SUPPORTED	0x00	Not supported
_NVME_MI_VPD_MAX_SMBUS_FREQ_100_KHZ	0x01	100 kHz
_NVME_MI_VPD_MAX_SMBUS_FREQ_400_KHZ	0x02	400 kHz
_NVME_MI_VPD_MAX_SMBUS_FREQ_1_MHZ	0x03	1 MHz

in.NvmeBasicManagement: Bit 0 in this field, if set to '1', indicates if the port implements the NVMe Basic Management command. All other bits in this field are reserved.

5.6.1.1.2.4.3 Controller Information Data Structure

in.PORTID: This field specifies the PCIe port identifier with which the Controller is associated.

in.PCIeRoutingIdValid: This bit is set to '1' if the device has captured a Bus Number and Device Number (Bus Number only for ARI devices). This bit is set to '0' if the device has not captured a Bus and Device number (Bus Number only for ARI devices).

in.PCIeBusNumber: The Controller's PCI Bus Number.

in.PCIeDeviceNumber: The Controller's PCI Bus Number.

in.PCIeFunctionNumber: The Controller's PCI Function Number.

in.PCIeVendorId: The PCI Vendor ID for the specified Controller.

in.PCIeDeviceId: The PCI Device ID for the specified Controller.

in.PCIeSubsystemVendorId: The PCI Subsystem Vendor ID for the specified Controller.

in.PCIeSubsystemDeviceId: The PCI Subsystem Device ID for the specified Controller.

5.6.1.1.2.4.4 Optionally Supported Command Data Structure

in.NUMCMD: This field contains the number of optionally supported commands in the list. A value of 0h indicates there are no commands in the list.

in.NMIMT: This field specifies the NVMe-MI Message Type.

in.OpCode: This field specifies the opcode used for the optionally supported command.

5.6.1.1.2.5 VPD Read

in.VPD: Vital Product Data .

5.6.1.1.3 NVM Express Admin Command Set

in.CID: Command Identifier. Indicates the identifier of the command that is being completed. Shall be cleared to 0h.

in.P: Phase Tag. Identifies whether a Completion Queue entry is new.

in.SC: Status Code. Indicates a status code identifying any error or status information for the command indicated.

in.SCT: Status Code Type. Indicates the status code type of the completion queue entry. This indicates the type of status the controller is returning.

in.M: More. If set to '1', there is more status information for this command as part of the Error Information log that may be retrieved with the Get Log Page command. If cleared to '0', there is no additional status information for this command.

in.DNR: Do Not Retry. If set to '1', indicates that if the same command is re-submitted it is expected to fail. If cleared to '0', indicates that the same command may succeed if retried.

5.6.1.1.4 Get Features

5.6.1.1.4.1 Arbitration (Feature Identifier 01h)

in.AB: Indicates the maximum number of commands that the controller may launch at one time from a particular Submission Queue.

in.LPW: This field defines the number of commands that may be executed from the low priority service class in each arbitration round. This is a 0's based value

in.MPW: This field defines the number of commands that may be executed from the medium priority service class in each arbitration round. This is a 0's based value.

in.HPW: This field defines the number of commands that may be executed from the high priority service class in each arbitration round. This is a 0's based value

5.6.1.1.4.2 Power Management (Feature Identifier 02h)

in.PS: This field indicates the new power state into which the controller should transition.

in.WH: This field indicates the new power state into which the controller should transition.

Constant	Code	Command
_NVME_MI_NVM_ADMIN_WH_NO_WORKLOAD	0x00	No Workload
_NVME_MI_NVM_ADMIN_WH_WORKLOAD_1	0x01	Workload #1
_NVME_MI_NVM_ADMIN_WH_WORKLOAD_2	0x02	Workload #2

5.6.1.1.4.3 LBA Range Type (Feature Identifier 03h)

in.NUM: This field specifies the number of LBA ranges specified in this command.

5.6.1.1.4.4 Temperature Threshold (Feature Identifier 04h)

in.TMPTH: Indicates the threshold value to apply (in a Set Features) or return (in a Get Features) for the temperature sensor and threshold type specified.

in.TMPSEL: This field selects the temperature whose threshold is modified by a Set Features command and whose threshold value is returned by a Get Features command.

Constant	Code	Command
_NVME_MI_NVM_ADMIN_COMPOSITE_TEMPERATURE	0x00	Composite Temperature
_NVME_MI_NVM_ADMIN_TEMPERATURE_SENSOR_1	0x01	Temperature Sensor 1
_NVME_MI_NVM_ADMIN_TEMPERATURE_SENSOR_2	0x02	Temperature Sensor 2
_NVME_MI_NVM_ADMIN_TEMPERATURE_SENSOR_3	0x03	Temperature Sensor 3
_NVME_MI_NVM_ADMIN_TEMPERATURE_SENSOR_4	0x04	Temperature Sensor 4
_NVME_MI_NVM_ADMIN_TEMPERATURE_SENSOR_5	0x05	Temperature Sensor 5
_NVME_MI_NVM_ADMIN_TEMPERATURE_SENSOR_6	0x06	Temperature Sensor 6
_NVME_MI_NVM_ADMIN_TEMPERATURE_SENSOR_7	0x07	Temperature Sensor 7
_NVME_MI_NVM_ADMIN_TEMPERATURE_SENSOR_8	0x08	Temperature Sensor 8
_NVME_MI_NVM_ADMIN_ALL_IMPLEMENTED	0x0F	All implemented temperature sensors in a Set Features command. Reserved in a Get Features command

in.THSEL: This field selects the threshold type that is modified by a Set Features command and whose threshold value is returned by a Get Features command.

Constant	Code	Command
_NVME_MI_NVM_ADMIN_OVER_TEMPERATURE_THRESHOLD	0x00	Over Temperature Threshold
_NVME_MI_NVM_ADMIN_UNDER_TEMPERATURE_THRESHOLD	0x01	Under Temperature Threshold

5.6.1.1.4.5 Error Recovery (Feature Identifier 05h)

in.TLER: Indicates a limited retry timeout value in 100 millisecond units.

in.DULBE: If set to '1', then the Deallocated or Unwritten Logical Block error is enabled for the namespace specified in CDW1.NSID.

5.6.1.1.4.6 Volatile Write Cache (Feature Identifier 06h)

in.WCE: If set to '1', then the volatile write cache is enabled. If cleared to '0', then the volatile write cache is disabled.

5.6.1.1.4.7 Number of Queues (Feature Identifier 07h)

in.NSQR: Indicates the number of I/O Completion Queues requested by software.

in.NCQR: Indicates the number of I/O Submission Queues requested by software.

5.6.1.1.4.8 Interrupt Coalescing (Feature Identifier 08h)

in.TIME: Specifies the recommended maximum time in 100 microsecond increments that a controller may delay an interrupt due to interrupt coalescing.

in.THR: Specifies the recommended minimum number of completion queue entries to aggregate per interrupt vector before signaling an interrupt to the host.

5.6.1.1.4.9 Interrupt Vector Configuration (Feature Identifier 09h)

in.CD: If set to '1', then any interrupt coalescing settings shall not be applied for this interrupt vector.

in.IV: This field specifies the interrupt vector for which the configuration settings are applied.

5.6.1.1.4.10 Write Atomicity Normal (Feature Identifier 0Ah)

in.DN: If set to '1', then the host specifies that AWUN and NAWUN are not required and that the controller shall only honor AWUPF and NAWUPF.

5.6.1.1.4.11 Asynchronous Event Configuration (Feature Identifier 0Bh)

in.SMART: This field determines whether an asynchronous event notification is sent to the host for the corresponding Critical Warning specified in the SMART / Health Information Log.

in.NamespaceAttributeNotices: This field determines whether an asynchronous event notification is sent to the host for a Namespace Attribute change.

in.FirmwareActivationNotices: This field determines whether an asynchronous event notification is sent to the host for a Firmware Activation Starting event.

5.6.1.1.4.12 Autonomous Power State Transition (Feature Identifier 0Ch)

in.APSTE: This field specifies whether autonomous power state transition is enabled.

5.6.1.1.4.13 Host Memory Buffer (Feature Identifier 0Dh)

in.GetFeatures_EHM: If set to '1', then the controller may use the host memory buffer. While cleared to '0', the controller shall not use the host memory buffer.

in.SetFeatures_EHM: If set to '1', then the controller may use the host memory buffer. While cleared to '0', the controller shall not use the host memory buffer.

in.GetFeatures_MR: If set to '1', then the host is returning previously allocated memory the controller used prior to a reset or entering the Runtime D3 state.

in.SetFeatures_MR: If set to '1', then the host is returning previously allocated memory the controller used prior to a reset or entering the Runtime D3 state.

in.GetFeatures_HSIZE: This field specifies the size of the host memory buffer allocated in memory page size (CC.MPS) units.

in.SetFeatures_HSIZE: This field specifies the size of the host memory buffer allocated in memory page size (CC.MPS) units.

in.GetFeatures_HMDLAL: This field specifies the lower 32 bits of the physical location of the Host Memory Descriptor List for the Host Memory Buffer.

in.SetFeatures_HMDLAL: This field specifies the lower 32 bits of the physical location of the Host Memory Descriptor List for the Host Memory Buffer.

in.GetFeatures_HMDLAU: This field specifies the upper 32 bits of the physical location of the Host Memory Descriptor List for the Host Memory Buffer.

in.SetFeatures_HMDLAU: This field specifies the upper 32 bits of the physical location of the Host Memory Descriptor List for the Host Memory Buffer.

in.GetFeatures_HMDLEC: This field specifies the number of entries provided in the Host Memory Descriptor List.

in.SetFeatures_HMDLEC: This field specifies the number of entries provided in the Host Memory Descriptor List.

5.6.1.1.4.14 Software Progress Marker (Feature Identifier 80h)

in.PBSLC: Indicates the load count of pre-boot software.

5.7 Lane Margining specific set of members

Valid for Lane Margining (LM) Level transactions only. Undefined for other levels.

in.CmdType : returns command types as integer for each lane in current transaction.

in.CmdTypeToStr : returns all possible command types as string array.

in.ErrorsCount : returns the amount of errors for current transaction.

in.ErrorsDescr : returns the description for all errors in current transaction.

in.IsControlSKP: returns 1 if this is Control SKIP, else returns 0.

in.ReceiverNumberToStr : returns all possible receiver numbers as strings.

in.RequestMarginType : returns Margin Type as integer for each lane in the request part of current transaction.

in.RequestMarginParity : returns Margin Parity as integer for each lane in the request part of current transaction.

in.RequestMarginPayload : returns Margin Payload as integer for each lane in the request part of current transaction.

in.RequestReceiverNumber : returns Receiver Number as integer for each lane in the request part of current transaction.

in.RequestUsageModel : returns Usage Model as integer for each lane in the request part of current transaction.

in.ResponseMarginType : returns Margin Type as integer for each lane in the response part of current transaction.

in.ResponseMarginParity : returns Margin Parity as integer for each lane in the response part of current transaction.

in.ResponseMarginPayload : returns Margin Payload as integer for each lane in the response part of current transaction.

in.ResponseReceiverNumber : returns Receiver Number as integer for each lane in the response part of current transaction.

in.ResponseUsageModel : returns Usage Model as integer for each lane in the response part of current transaction.

in.UsedLanesCount : returns the amount of lanes used in this transaction.

in.UsedLanesIndexes : returns Completion Status for Split in LN Read transaction.

5.8 Lightweight Notification specific set of members

Valid for Ligthweight Notification (LN) Level transactions only. Undefined for other levels.

in.CLS : returns cacheline size : 64 or 128.

in.CompletionStatus: returns Completion Status for Split in LN Read transaction.

in.HasLnMessage : returns 1 if this transaction contains LN Message.

in.IsIncomplete : returns 1 if this transaction does not contain LN Message.

in.IsLNMessageBroadcast : returns 1 if this transaction contains broadcast LN Message.

in.IsLNRead : returns 1 if this is LN Read transaction.

in.NR : returns -1 if this message does not contain LN Message. Returns the value of Notification Reason (NR) field :

Constant	Code	Command
_LN_NR_CACHELINE_UPDATE	0x00	LN Message was sent due to a cacheline update
_LN_NR_EVICTION_SINGLE	0x01	LN Message was sent due to the eviction of a single cacheline
_LN_NR_EVICTION_ALL	0x02	LN Message was sent due to the eviction of all cachelines registered to this Function
_LN_NR_RESERVED	0x03	Reserved

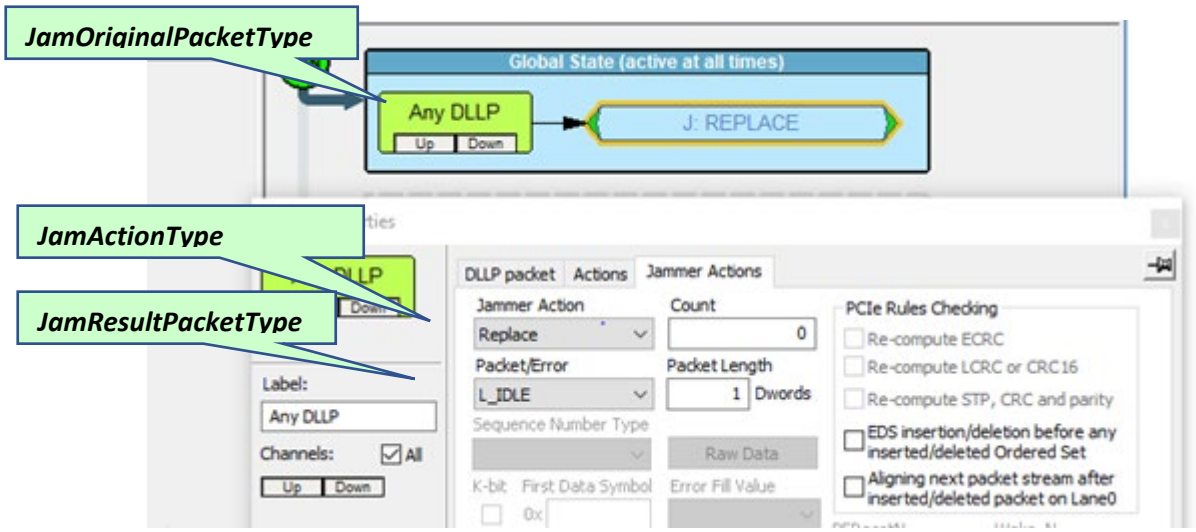
5.9 Verification Script Engine Input Context Members

Input context was extended with new fields for traces with jammed traffic.
in. JamActionType: Contains the numeric encoding of the Jammer Action type. The following possible values are defined by VSE and the corresponding constants can be used by scripts:

```
JAMMER_DELETE            = 81;
JAMMER_REPLACE           = 82;
JAMMER_MODIFY            = 83;
JAMMER_INS_PKT_X_AFTER   = 84;
JAMMER_INS_ERROR         = 85;
JAMMER_SIDEBAND          = 86;
```

in. JamOriginalPacketType: String representation of event type
in. JamResultPacketType: String representation of result packet type (valid for jammer Replace, Insert Pkt X after, insert error).

New input context fields have the same meaning and behavior as corresponding settings in Event Properties dialogue:



5.10 CXL Specific Set of Members

5.10.1 CXL Packet's Flit Generic Fields

in.CXL_ProtocolId: Protocol ID value of the flit. Applicable for CXL 68B mode only.

in.CXL_ProtocolIdStr: Protocol ID value of the flit. Returns string. Applicable for CXL 68B mode only.

in.CXL_FlitHasImpliedEDS: if flit has Implied EDS. Returns 0 or 1. Applicable for CXL 68B mode only.

in.CXL_HasSyncBits: if flit has Sync Header. Returns 0 or 1. Applicable for CXL 68B mode only.

5.10.1.1 Errors

in.CXL_FlitErrorInvalidProtocolId: if flit has Invalid Protocol ID error. Returns 0 or 1. Applicable for CXL 68B mode only.

in.CXL_FlitErrorReservedNotZero: if flit has reserved bytes not zero. Returns 0 or 1. Applicable for CXL 68B mode only.

in.CXL_FlitErrorBadCRC: if flit has bad CRC. Returns 0 or 1.

in.CXL_ControlFlitErrorFormat: if CXL.cache/mem Control flit has error in format. Returns 0 or 1. Applicable for CXL 68B mode only.

5.10.2 CXL ALMP Packet's Flit Fields

in.CXL_ALMPMessage: ALMP packet Message field.

in.CXL_ALMPType: ALMP packet Type field.

in.CXL_ALMPTypeStr: ALMP packet Type field. Returns string.

in.CXL_ALMPVLSMState: ALMP packet VLSM State field.

in.CXL_ALMPVLSMStateStr: ALMP packet VLSM State field. Returns string.

in.CXL_ALMPTarget: ALMP packet VLSM Target field.

in.CXL_ALMPTargetStr: ALMP packet VLSM Target field. Returns string.

5.10.2.1 Errors

in.CXL_ALMPIntegrityError: if ALMP flit has integrity error. Returns 0 or 1.

in.CXL_ALMPInvalidMessageCode: if ALMP flit has invalid message code field. Returns 0 or 1.

in.CXL_ALMPReservedIsNotZero: if ALMP flit reserved bits are not zero. Returns 0 or 1.

in.CXL_ALMPStateEncodingError: if ALMP flit state has encoding error. Returns 0 or 1.

in.CXL_ALMPInstanceNumberError: if ALMP flit has instance number error. Returns 0 or 1.

5.10.3 CXL Cache/Mem Packet's Flit Header Fields

in.CXL_FlitCRC: CXL.cache/mem flit CRC value.

in.CXL_CacheMemFlitType: CXL.cache/mem flit type (Control or Protocol).

in.CXL_CacheMemFlitRspCrd: CXL.cache/mem flit response credits value.

in.CXL_CacheMemFlitReqCrd: CXL.cache/mem flit request credits value.

in.CXL_CacheMemFlitDataCrd: CXL.cache/mem flit data credits value.

in.CXL_CacheMemFlitDataRollover: CXL.cache/mem flit data rollover.

in.CXL_CacheMemAllDataFlit: if CXL.cache/mem flit is All-Data flit. Returns 0 or 1.

in.CXL_CacheMemPacketSlotType: cache/mem packet Slot value defined in protocol flit header. In case of all data flit where flit header is absent, this field is set to 0.

in.CXL_CacheMemPacketSlotTypeStr: field returns the string with packet slot info. Slot info contains packet slot number in protocol flit and decoded packet slot value (according to Table 50. Slot Format field Encoding, CXL 2.0 spec). Slot info string format "Slotn : SlotName" where n is packet slot number, SlotName - decoded slot value. In case of all data flits where flit header is absent, SlotName is set to "G0" and slot number is calculated based on packet offset in flit.

5.10.4 CXL Cache/Mem Protocol Packet's Fields

in.CXL_CacheMemPacketValid: CXL.cache/mem protocol packet Valid field. Returns 0 or 1. If packet is IDE encrypted and decryption has been finished with errors or transaction for which packet belongs to is not acknowledged or not valid or Valid field is not set, Valid field returns -1.

in.CXL_CacheMemPacketOpCode: CXL.cache/mem protocol packet OpCode field. If packet is IDE encrypted and decryption has finished with errors or packet transaction is not acknowledged or not valid or OpCode field is not set, OpCode field returns -1.

in.CXL_CacheMemPacketPoison: CXL.cache/mem protocol packet Poison field. If packet is IDE encrypted and decryption has finished with errors or packet transaction is not acknowledged or not valid or Poison field is not set, Poison field returns -1.

in.CXL_MemPacketMetaField: CXL.mem protocol packet MetaField field. If packet is IDE encrypted and decryption has finished with errors or packet transaction is not acknowledged or not valid or MetaField field is not set, MetaField field returns -1.

in.CXL_MemPacketMetaValue: CXL.mem protocol packet MetaValue field. If packet is IDE encrypted and decryption has finished with errors or packet transaction is not acknowledged or not valid or MetaValue field is not set, MetaValue field returns -1.

in.CXL_MemPacketSnpType: CXL.mem protocol packet Snoop Type field. If packet is IDE encrypted and decryption has finished with errors or packet transaction is not acknowledged or not valid or SnoopType field is not set, Snoop Type field returns -1.

in.CXL_CacheMemPacketAddressLow: CXL.mem protocol packet Address field low part (32 bits).

in.CXL_CacheMemPacketAddressHi: CXL.mem protocol packet Address field high part (32 bits).

in.CXL_MemPacketTC: CXL.mem protocol packet Traffic Class field. If packet is IDE encrypted and decryption has finished with errors or packet transaction is not acknowledged or not valid or Traffic Class field is not set, Traffic Class field returns -1.

in.CXL_H2DResponseRspData: CXL.cache H2D Response RspData field. If packet is IDE encrypted and decryption has finished with errors or packet transaction is not acknowledged or not valid or RspData field is not set, RspData field returns -1.

in.CXL_H2DResponseRSP_PRE: CXL.cache H2D Response RSP_PRE field. If packet is IDE encrypted and decryption has finished with errors or packet transaction is not acknowledged or not valid or RSP_PRE field is not set, RSP_PRE field returns -1.

in.CXL_H2DDataHeaderGoErr: CXL.cache H2D Data Header GoErr field. If packet is IDE encrypted and decryption has finished with errors or packet transaction is not acknowledged or not valid or GoErr field is not set, GOErr field returns -1.

in.CXL_CachePacketCQID: CXL.cache protocol packet CQID field. If packet is IDE encrypted and decryption has finished with errors or packet transaction is not acknowledged or not valid or CQID field is not set, CQID field returns -1.

in.CXL_CachePacketUQID: CXL.cache protocol packet UQID field. If packet is IDE encrypted and decryption has finished with errors or packet transaction is not acknowledged or not valid or UQID field is not set, UQID field returns -1.

in.CXL_MemPacketTag: CXL.mem protocol packet Tag field. If packet is IDE encrypted and decryption has finished with errors or packet transaction is not acknowledged or not valid or Tag field is not set, Tag field returns -1.

in.CXL_DataHeaderChunkValid: CXL.cache/mem Data Header Chunk Valid field. If packet is IDE encrypted and decryption has finished with errors or packet transaction is not acknowledged or not valid or Chunk Valid field is not set, Chunk Valid field returns -1.

in.CXL_D2HDataHeaderBogus: CXL.cache/mem Data Header Bogus field. If packet is IDE encrypted and decryption has finished with errors or packet transaction is not acknowledged or not valid or Data Header Bogus field is not set, Data Header Bogus field returns -1.

in.CXL_MemPacketLDID: CXL.mem protocol packet LDID field. If packet is IDE encrypted and decryption has finished with errors or packet transaction is not acknowledged or not valid or LDID field is not set, LDID field returns -1.

in.CXL_MemPacketDevLoad: CXL.mem protocol packet DevLoad field. If packet is IDE encrypted and decryption has finished with errors or packet transaction is not acknowledged or not valid or DevLoad field is not set, DevLoad field returns -1.

in.IsCxlCacheMemIdeEncrypted: true if packet is IDE encrypted, false otherwise.

in.IsCxlCacheMemIdeDecrypted: true if packet is IDE encrypted and decryption has been finished without errors. If packet is not IDE encrypted or decryption has been finished with errors or transaction for which packet belongs to is not acknowledged or not valid, this field is equal to false.

5.10.4.1 Errors

in.CXL_ErrorCacheMemReservedIsNotZero: if CXL.cache/mem protocol flit has reserved bits not zero. Returns 0 or 1.

in.CXL_ErrorCacheMemOpcode: if CXL.cache/mem protocol packet Opcode is invalid. Returns 0 or 1.

in.CXL_ErrorCacheMemMetaField: if CXL.cache/mem protocol packet Meta field is invalid. Returns 0 or 1.

in.CXL_ErrorCacheMemMetaValue: if CXL.cache/mem protocol packet Meta value is invalid. Returns 0 or 1.

in.CXL_ErrorCacheMemSnoopType: if CXL.cache/mem protocol packet Snoop value is invalid. Returns 0 or 1.

in.CXL_ErrorCacheMemTCIsNotZero: if CXL.cache/mem protocol packet TC (Traffic Class) is not zero. Returns 0 or 1.

in.CXL_ErrorCacheMemAddress5IsNotZero: if CXL.mem protocol packet Address field bit 5 is not zero. Returns 0 or 1.

5.10.5 CXL Cache/Mem Control Packet's Fields

in.CXL_CacheMemControlFlitType: CXL.cache/mem control flit type (RETRY, LLCRD, IDE, etc).

in.CXL_CacheMemControlFlitTypeStr: CXL.cache/mem control flit type. Returns string.

in.CXL_CacheMemControlFlitSubType: CXL.cache/mem control flit sub type (RETRY.Idle, RETRY.Req, LLCRD.Ack, etc).

in.CXL_CacheMemControlFlitSubTypeStr: CXL.cache/mem control flit sub type. Returns string.

in.CXL_CacheMemControlFull_Ack: CXL.cache/mem control LLCRD packet FullAck field.

in.CXL_CacheMemControlEseq: CXL.cache/mem control retry packet Eseq field.

in.CXL_CacheMemControlNumRetry: CXL.cache/mem control retry packet NumRetry field.

in.CXL_CacheMemControlNumPhyReinit: CXL.cache/mem control retry packet NumPhyReinit field.

in.CXL_CacheMemControlEmpty: CXL.cache/mem control packet retry Ack Empty field.

in.CXL_CacheMemControlViral: CXL.cache/mem control packet retry Ack Viral field.

in.CXL_CacheMemControlWrPtr: CXL.cache/mem control packet retry Ack WrPtr field.

in.CXL_CacheMemControlNumFreeBuf: CXL.cache/mem control packet retry ack NumPhyReinit field.

in.CXL_CacheMemControlViralLDIDVector: CXL.cache/mem control packet retry ack Viral LDID Vector field.

in.CXL_CacheMemControlMacField_7_0: CXL.cache/mem control IDE packet MAC field bits 0:7.

in.CXL_CacheMemControlMacField_31_8: CXL.cache/mem control IDE packet MAC field bits 8:31.

in.CXL_CacheMemControlMacField_63_32: CXL.cache/mem control IDE packet MAC field bits 32:63.

in.CXL_CacheMemControlMacField_95_64: CXL.cache/mem control IDE packet MAC field bits 64:95.

in.CXL_CacheMemControlInterconnectVersion: CXL.cache/mem control Init param packet InterconnectVersion field.

in.CXL_CacheMemControlLLRWrapValue: CXL.cache/mem control Init Param packet LLRWrapValue field.

in.CXL_CacheMemFlitCltFmt: CTL_FMT field value of Link Layer Control Flit header.
NOTE: The CTL_FMT field values are reserved in CXL 1.1. However, to be consistent with later specs (i.e., CXL 2.0, CXL 3.0), this function will return 0 if it is called for in flits in a CXL 1.1 trace.

5.10.5.1 Errors

in.CXL_ErrorCtrlFlitReservedIsNotZero: if CXL.cache/mem control flit has reserved bits not zero. Returns 0 or 1.

in.CXL_ErrorCtrlFlitType: if CXL.cache/mem control flit type is invalid. Returns 0 or 1.

in.CXL_ErrorCtrlFlitSubType: if CXL.cache/mem control flit sub-type is invalid. Returns 0 or 1.

in.CXL_ErrorCtrlFlitNumFreeBuf: if CXL.cache/mem control flit NumFree value is invalid. Returns 0 or 1.

in.CXL_ErrorCtrlFlitInterconnectVer: if CXL.cache/mem control flit Interconnect Version value is invalid. Returns 0 or 1.

5.10.6 CXL.io Vendor Defined Message PM TLP's Fields

in.IsCXLPMVDM: if TLP is CXL VDM Power Message (has CXL Vendor ID).

in.MessageCXLOpCode: CXL VDM TLP PM OpCode field value.

in.MessageCXLOpCodeStr: CXL VDM TLP PM OpCode field value. Returns string.

in.MessageCXLResetType: CXL VDM TLP PM Reset Type value (if OpCode is correspondent).

in.MessageCXLPrepType: CXL VDM TLP PM Prep Type value (if OpCode is correspondent).

in.IsCXLVDMRequest: if CXL VDM TLP PM is Request. Returns 0 or 1.

in.CXLVDMGPFPhaseValue: CXL VDM TLP PM GPF Phase value (for correspondent VDM TLP).

in.CXLFwInterruptVector: CXL VDM TLP PM FwInterrupt value (for correspondent VDM TLP).

in.CXLVDMGPFType: CXL VDM TLP PM GPF Type Value (for correspondent VDM TLP)

in.CXLVDMGPFStatus: CXL VDM TLP PM GPF Status Value (for correspondent VDM TLP)

in.CXLVDMGPFPowerfail: CXL VDM TLP PM GPF Powerfail Value. Indicates powerfail is imminent. Returns 0 or 1.

in.CXLVDMGPFFlush : CXL VDM TLP PM GPF Flush Value. Indicates device must flush its caches. Returns 0 or 1.

in.CXLVDMGPFCacheFlushError: CXL VDM TLP PM Cache Flush Error Value. Indicates Cache Flush phase encountered an error. Returns 0 or 1.

5.10.7 CXL Data Transaction-Specific Set of Members

Valid for CXL Data transactions only, undefined for other events.

in.Payload: CXL Data transaction payload bytes array.

5.10.8 CXL.cache/mem Link Transaction-Specific Set of Members

Valid for CXL.cache/mem Link transactions only, undefined for other events.

in.CXL_CacheMemLinkTraPacketType: CXL.cache/mem Link Transaction packet type field.

in.CXL_CacheMemLinkTraAddress: CXL.cache/mem Link Transaction address field.

in.CXL_CacheLinkTraCQID: CXL.cache/mem Link Transaction CQID field.

in.CXL_CacheLinkTraUQID: CXL.cache/mem Link Transaction UQID field.

in.CXL_CacheLinkTraTag: CXL.cache/mem Link Transaction Tag field.

in.CXL_CacheLinkTraStatus: CXL.cache/mem Link Transaction status field.

5.10.9 CXL.cache Split Transaction-Specific Set of Members

Valid for CXL.cache Split transactions only, undefined for other events.

in.CXL_CacheMemSplitTraPacketType: CXL.cache Split Transaction packet type field.

in.CXL_CacheMemSplitTraAddress: CXL.cache Split Transaction address field.

in.CXL_CacheSplitTraCQID: CXL.cache Split Transaction CQID field.

in.CXL_CacheSplitTraUQID: CXL.cache Split Transaction UQID field.

in.CXL_CacheSplitTraStatus: CXL.cache Split Transaction status field.

5.10.10 CXL.mem Split Transaction-Specific Set of Members

Valid for CXL.mem Split transactions only, undefined for other events.

in.CXL_CacheMemSplitTraPacketType: CXL.mem Split Transaction packet type field.

in.CXL_CacheMemSplitTraAddress: CXL.mem Split Transaction address field.

in.CXL_MemSplitTraTag: CXL.mem Split Transaction Tag field.

in.CXL_MemSplitTraStatus: CXL.mem Split Transaction status field.

5.11 DOE Transaction-Specific Set of Members

in.IsVendorIDAvailable - Is VendorID available

in.VendorID - Vendor ID

Supported Vendor IDs:
_PCI_SIG_ID - 0x0001
_CXL_ID - 0x1E98

in.DOEPayloadLength - DOE Payload length in dwords

in.DOEPayload - DOE Payload

in.IsDataObjectTypeAvailable - Is DataObjectType available

in.DataObjectType - Data object type

Supported DataObjectTypes for PCI_SIG Vendor (_PCI_SIG_ID - 0x0001):
_DOE_DISCOVERY_ID - 0
_DOE_SPDM_ID - 1
_DOE_SECURED_SPDM_ID - 2

Supported DataObjectTypes for CXL Vendor (_CXL_ID - 0x1E98):
_DOE_PROTOCOL_CXL_COMPLIANCE_ID - 0
_DOE_PROTOCOL_CXL_TABLEACCESS_ID - 2

in.IsDOEPayloadLengthFromHeaderAvailable - Is DOEPayloadLengthFromHeader available

in.DOEPayloadLengthFromHeader - DOE Payload length in dwords (value from header if available)

If no errors: in.DOEPayloadLengthFromHeader = in.DOEPayloadLength

in.IsDataObjectRequest - Is DOE request

in.IsDataObjectResponse - Is DOE response

in.IsDataObjectDiscovery - Is DOE Discovery

in.DiscoveryIndex - Discovery Index

in.DiscoveryVendorID - Discovery VendorID

in.DiscoveryDataObjectProtocol - Discovery DataObjectProtocol

in.DiscoveryNextIndex - Discovery NextIndex

in.IsAborted - Is aborted DOE transaction (DOE transaction with errors)

5.12 SPDM Transaction-specific Set of Members

in.IsDecodingSupported: returns 1 if RequestResponseCode-specific fields can be read for this message

in.IsRequest: returns 1 if this message is request

in.HasErrors: returns 1 if this message has one or more SPDM or Secured SPDM errors

in.Transport: returns transport protocol name: “DOE” or “MCTP”

in.IsSecured: returns 1 if the protocol id is Secured SPDM

in.IsPayloadEncrypted: returns 1 if payload is encrypted and decrypted payload can’t be accessed

in.IsHeaderPresent: returns 1 if SPDM header is present and can be read

in.SPDMVersion: returns SPDMVersion field value

in.RequestCode: returns RequestResponseCode field value for requests or RequestCode value for encapsulated messages

in.ResponseCode: returns RequestResponseCode field value for responses or ResponseCode value for encapsulated messages

in.Param1: returns Param1 field value

in.Param2: returns Param2 field value

in.LengthError: returns 1 if the error is present

in.ReservedValueError: returns 1 if the error is present

in.ReservedNotZeroError: returns 1 if the error is present

in.MissingDataError: returns 1 if the error is present

in.AmbiguousAlgorithmsError: returns 1 if the error is present

in.IllegalCapabilitiesError: returns 1 if the error is present

in.NotAllowedError: returns 1 if the error is present

in.NotAllowedEncapsulatedError: returns 1 if the error is present

in.MissingKeyError: returns 1 if the error is present

in.StructureError: returns 1 if the error is present

in.SecuredLengthError: returns 1 if the error is present

in.ApplicationDataLengthError: returns 1 if the error is present

in.IntegrityError: returns 1 if the error is present

in.OutOfSequenceError: returns 1 if the error is present

in.VendorDefinedPayload: returns vendor-defined payload for VENDOR_DEFINED_REQUEST and VENDOR_DEFINED_RESPONSE messages as an array of bytes

in.VendorDefinedPayload_field_size: returns the size of VendorDefinedPayload array

in.VendorDefinedPayloadLength: returns ReqLength/RespLength field value for VENDOR_DEFINED_REQUEST/VENDOR_DEFINED_RESPONSE messages

The following keys can only be used for NEGOTIATE_ALGORITHMS and ALGORITHMS messages:

in.BaseHash_TPM_ALG_SHA_256: returns 1 if the device indicates support of the algorithm

in.BaseHash_TPM_ALG_SHA_384: returns 1 if the device indicates support of the algorithm

in.BaseHash_TPM_ALG_SHA_512: returns 1 if the device indicates support of the algorithm

in.BaseHash_TPM_ALG_SHA3_256: returns 1 if the device indicates support of the algorithm

in.BaseHash_TPM_ALG_SHA3_384: returns 1 if the device indicates support of the algorithm

in.BaseHash_TPM_ALG_SHA3_512: returns 1 if the device indicates support of the algorithm

in.BaseHash_TPM_ALG_SM3_256: returns 1 if the device indicates support of the algorithm

in.ffdhe2048: returns 1 if the device indicates support of the algorithm

in.ffdhe3072: returns 1 if the device indicates support of the algorithm

in.ffdhe4096: returns 1 if the device indicates support of the algorithm

in.secp256r1: returns 1 if the device indicates support of the algorithm

in.secp384r1: returns 1 if the device indicates support of the algorithm

in.secp521r1: returns 1 if the device indicates support of the algorithm

in.SM2_P256: returns 1 if the device indicates support of the algorithm

in.AES_128_GCM: returns 1 if the device indicates support of the algorithm

in.AES_256_GCM: returns 1 if the device indicates support of the algorithm

in.CHACHA20_POLY1305: returns 1 if the device indicates support of the algorithm

in.AEAD_SM4_GCM: returns 1 if the device indicates support of the algorithm

The following keys can only be used for ALGORITHMS message:

in.BaseAsym_TPM_ALG_RSASSA_2048: returns 1 if the device indicates support of the algorithm

in.BaseAsym_TPM_ALG_RSAPSS_2048: returns 1 if the device indicates support of the algorithm

in.BaseAsym_TPM_ALG_RSASSA_3072: returns 1 if the device indicates support of the algorithm

in.BaseAsym_TPM_ALG_RSAPSS_3072: returns 1 if the device indicates support of the algorithm

in.BaseAsym_TPM_ALG_ECDSA_ECC_NIST_P256: returns 1 if the device indicates support of the algorithm

in.BaseAsym_TPM_ALG_RSASSA_4096: returns 1 if the device indicates support of the algorithm

in.BaseAsym_TPM_ALG_RSAPSS_4096: returns 1 if the device indicates support of the algorithm

in.BaseAsym_TPM_ALG_ECDSA_ECC_NIST_P384: returns 1 if the device indicates support of the algorithm

in.BaseAsym_TPM_ALG_ECDSA_ECC_NIST_P521: returns 1 if the device indicates support of the algorithm

in.BaseAsym_TPM_ALG_SM2_ECC_SM2_P256: returns 1 if the device indicates support of the algorithm

in.BaseAsym_EdDSA_ed25519: returns 1 if the device indicates support of the algorithm

in.BaseAsym_EdDSA_ed448: returns 1 if the device indicates support of the algorithm

in.ReqBaseAsym_TPM_ALG_RSASSA_2048: returns 1 if the device indicates support of the algorithm

in.ReqBaseAsym_TPM_ALG_RSAPSS_2048: returns 1 if the device indicates support of the algorithm

in.ReqBaseAsym_TPM_ALG_RSASSA_3072: returns 1 if the device indicates support of the algorithm

in.ReqBaseAsym_TPM_ALG_RSAPSS_3072: returns 1 if the device indicates support of the algorithm

in.ReqBaseAsym_TPM_ALG_ECDSA_ECC_NIST_P256: returns 1 if the device indicates support of the algorithm

in.ReqBaseAsym_TPM_ALG_RSASSA_4096: returns 1 if the device indicates support of the algorithm

in.ReqBaseAsym_TPM_ALG_RSAPSS_4096: returns 1 if the device indicates support of the algorithm

in.ReqBaseAsym_TPM_ALG_ECDSA_ECC_NIST_P384: returns 1 if the device indicates support of the algorithm

in.ReqBaseAsym_TPM_ALG_ECDSA_ECC_NIST_P521: returns 1 if the device indicates support of the algorithm

in.ReqBaseAsym_TPM_ALG_SM2_ECC_SM2_P256: returns 1 if the device indicates support of the algorithm

in.ReqBaseAsym_EdDSA_ed25519: returns 1 if the device indicates support of the algorithm

in.ReqBaseAsym_EdDSA_ed448: returns 1 if the device indicates support of the algorithm

in.MeasurementHash_RawBitStreamOnly: returns 1 if the device indicates support of the algorithm

in.MeasurementHash_TPM_ALG_SHA_256: returns 1 if the device indicates support of the algorithm

in.MeasurementHash_TPM_ALG_SHA_384: returns 1 if the device indicates support of the algorithm

in.MeasurementHash_TPM_ALG_SHA_512: returns 1 if the device indicates support of the algorithm

in.MeasurementHash_TPM_ALG_SHA3_256: returns 1 if the device indicates support of the algorithm

in.MeasurementHash_TPM_ALG_SHA3_384: returns 1 if the device indicates support of the algorithm

in.MeasurementHash_TPM_ALG_SHA3_512: returns 1 if the device indicates support of the algorithm

in.MeasurementHash_TPM_ALG_SM3_256: returns 1 if the device indicates support of the algorithm

in.RawMessage: returns full message data as transferred but without padding that may be used by the transport protocol

in.RawMessageSize: returns size in bytes of RawMessage

in.PlainMessage: returns data of either plain SPDM message or decrypted SPDM message (without Secured SPDM header, random data, MAC and transport padding)

in.PlainMessageSize: returns size in bytes of PlainMessage

The rest of the implemented keys can be found in the example script examp_spdm.pevs. These general rules apply:

1) Keys for RequestResponseCode-specific fields are available only if in.IsDecodingSupported is true.
2) Keys for fields with size <= 4 bytes return single numeric values.
3) Keys for data fields (e.g. Signature, RequesterVerifyData etc.) return byte arrays.
4) "_field_size" ending can be used to get size of a certain field of array type. Syntax: in.<ArrayFieldName>_field_size.
5) Keys for fields which can be present more than once in a single message (e.g. VersionEntry for VERSION response, Digest for DIGESTS response) return arrays of fields. "_count" ending can be used to get the number of fields in the array (e.g. in.Digest_count returns the number of Digests in DIGESTS response message). Use [] operator to get field by index (e.g. in.Digest[0]).
6) Using "_field_size" ending for arrays of fields returns array of sizes of the corresponding fields. Use [] operator to get field size by index (e.g. in.Digest_field_size[0] returns the size of the first digest).

6 Verification Script Engine Output Context Members

All verification scripts have output contexts – some special structures whose members are filled by the script and can be used inside of the application. (For more details about output contexts, please refer to the *File-Based Decoding (FBD) Manual.*) The verification script output contexts have only one member:

out.Result: Result of the whole verification program defined in the verification script.

This member is supposed to have the values:
_VERIFICATION_PROGRESS (set by default when script starts running)
_VERIFICATION_PASSED
_VERIFICATION_FAILED
_VERIFICATION_PASSED_WARNING
_VERIFICATION_NOT_APPLICABLE

The last two values should be set if you decide that the recorded trace does (or does not) satisfy the imposed verification conditions. In both cases, the verification script stops running.

If you don't specify any of those values, the result of script execution is set as **_VERIFICATION_FAILED** at exit.

Note: If you don’t care about the results of the script that’s running, please call function ScriptForDisplayOnly() one time before stopping the script. Then the results are **DONE.**

7 Verification Script Engine Events

VSE defines a large group of trace “events” – on packet, link, split, AHCI, ATA, NVM transaction and NVM command levels – that can be passed to a verification script for evaluation or retrieving and displaying some contained information. The information about the type of event can be seen in **in.TraceEvent**. Please refer to the topic "Sending Functions" in this manual for details about how to specify transaction levels and which events should be sent to verification scripts.

7.1 Packet level events

The table below describes the current list of Packet level events (transaction level: 0) and value of **in.TraceEvent:**

Types of Packets	**in.TraceEvent**
Data Link Layer Packets (DLLP)	_PKT_DLLP
Transaction Layer Packets (TLP)	_PKT_TLP
Ordered Sets	_PKT_ORDERED_SET
Link Conditions	_PKT_LINK_CONDITION
System Management Bus (SMBus) packets	_PKT_SMBUS
PMUX Packets	_PKT_PMUX
Electrical Idle Enter	_PKT_EIE_EVENT
Invalid Packet	_PKT_INVALID
PCIe Flit Record	_PKT_PCIE_FLIT
CXL Null Flit	_PKT_CXL_NULL_FLIT
CXL ALMP	_PKT_CXL_ALMP
CXL Cache/Mem	_PKT_CXL_CACHE_MEM
CXL Implied EDS	_PKT_CXL_IMPLIED_EDS
CXL.io Flit with IDLEs	_PKT_CXL_IO_IDLE_FLIT

7.2 Link Transaction Level Events

The table below describes the current list of Link Transaction events (transaction level: 1) and the value of **in.TraceEvent**:

Types of Link Transactions	**in.TraceEvent**
Memory transactions	_LINK_MEMORY
IO transactions	_LINK_IO
Configuration transactions	_LINK_CONFIG
Message transactions	_LINK_MESSAGE
Completion transactions	_LINK_COMPLETION
Atomic transactions	_LINK_ATOMICOP
Invalid transactions	_LINK_INVALID_TLP
CXL.cache/mem transactions	_LINK_CXL_CACHE_MEM

7.3 Split Transaction Level Events

The table below describes the current list of Split Transaction events (transaction level: 2) and the value of **in.TraceEvent:**

Types of Split Transactions	**in.TraceEvent**
Memory transactions	_SPLIT_MEMORY
IO transactions	_SPLIT_IO
Configuration transactions	_SPLIT_CONFIG
CXL.cache transactions	_SPLIT_CXL_CACHE
CXL.mem transactions	_SPLIT_CXL_MEM

7.4 NVM Transaction level events

NVME level introduces own events. The table below describes the current list of NVM Transaction events (transaction level: 3) and value of **in.TraceEvent:**

Types of NVM Transactions	**in.TraceEvent**
Transactions that read or write controller register	_NVME_CONTROLLER_REG
Transactions that write doorbell register.	_NVME_DOORBELL_REG
Transactions that transfer admin submission command	_NVME_ADMIN_SUBMISSION_CMD
Transactions that transfer NVM submission command	_NVME_NVM_SUBMISSION_CMD
Transactions that transfer admin completion command	_NVME_ADMIN_COMPLETION_CMD
Transactions that transfer NVM completion command	_NVME_NVM_COMPLETION_CMD
Transactions that target idx/dat registers (registers that are at fixed offset from BAR2)	_NVME_IDX_DAT_REG
Transactions that target data which is referenced by PRP or SGL	_NVME_TRANSFERED_DATA
Generated for all PRP list transactions	_NVME_PRP
Generated for all SGL descriptors transactions	_NVME_SGL

7.5 NVM Command level events

NVM Command level introduces own events. The table below describes the current list of NVM Command events (transaction level:) and value of **in.TraceEvent**

Types of NVM Commands	**in.TraceEvent**
Admin command	_NVMC_ADMIN_COMMAND
NVM command	_NVMC_NVM_COMMAND
Security command	_NVMC_SECURITY_COMMAND
Flush command	_NVMC_FLUSH
Write command	_NVMC_WRITE
Read command	_NVMC_READ
Write Uncorrectable command	_NVMC_WRITE_UNCORRECTABLE
Compare command	_NVMC_COMPARE
Write Zeros command	_NVMC_WRITE_ZEROES
Dataset Management command	_NVMC_DATASET_MGMT
Reservation Register command	_NVMC_RESERVATION_REGISTER
Reservation Report command	_NVMC_RESERVATION_REPORT
Reservation Acquire command	_NVMC_RESERVATION_ACQUIRE
Reservation Release command	_NVMC_RESERVATION_RELEASE
Delete I/O Submission Queue command	_NVMC_DELETE_IO_SQ

Types of NVM Commands	in.TraceEvent
Create I/O Submission Queue command	_NVMC_CREATE_IO_SQ
Get Log Page command	_NVMC_GET_LOG_PAGE
Delete I/O Completion Queue command	_NVMC_DELETE_IO_CQ
Create I/O Completion Queue command	_NVMC_CREATE_IO_CQ
Identify command	_NVMC_IDENTIFY
Abort command	_NVMC_ABORT
Set Feature command	_NVMC_SET_FEATURE
Get Feature command	_NVMC_GET_FEATURE
Asynchronous Event Request command	_NVMC_ASYNC_EVENT_REQUEST
Firmware Activate command	_NVMC_FIRMWARE_ACTIVATE
Firmware Image Download command	_NVMC_FIRMWARE_IMG_DOWNLOAD
Format NVM command	_NVMC_FORMAT_NVM
Security Send command	_NVMC_SECURITY_SEND
Security Receive command	_NVMC_SECURITY_RECEIVE

Note：To access PRP data from IO Commands you can use the following packet/transaction-specific set of members:

in.Payload: Bit source of the frame/sequence payload (you can extract any necessary information using the **GetNBits(), NextNBits(),** or **PeekNBits()** functions.

Note: in.Payload does not support SCSI.

in.PayloadLength: Length (in bytes of the retrieved payload)

7.6 AHCI Transaction level events

The table below describes the current list of AHCI Transaction events (transaction level: 5) and value of **in.TraceEvent:**

Types of AHCI Transactions	in.TraceEvent
Transactions that use HBA registers	_AHCI_HBA_REG
Transactions that use Ports registers	_AHCI_PORT_REG
Transactions that use Command list registers	_AHCI_CMND_LIST
Transaction uses FIS registers	_AHCI_RECEIVED_FIS
Transaction uses Command table registers	_AHCI_CMND_TABLE

7.7 ATA Transaction level events

ATA level introduces no events. Use SendAllTraceEvents() function to get ATA events.

7.8 CXL Data Level Events

Type of CXL Data Level Transaction	in.TraceEvent
CXL.cache/mem Data Transaction	_PKT_CXL_DATA_CACHE_MEM

8 Sending Functions

This topic contains information about the special group of VSE functions designed to specify which events the verification script should expect to receive.

8.1 SendLevel()

This function specifies that events of the specified transaction level should be sent to the script.

Format: **SendLevel(level)**

Level	Value	Description
_PACKET	(value 0)	Send Packet level events
_LINK	(value 1)	Send Link Transaction level events
_SPLIT	(value 2)	Send Split Transaction level events
_NVME	(value 3)	Send NVM Transaction level events
_NVMC	(value 9)	Send NVM Command level events
_AHCI	(value 5)	Send AHCI Transaction level events
_ATA	(value 6)	Send ATA Transaction level events
_PQI	(value 4)	Send PQI Transaction level events
_SOP	(value 7)	Send SOP Transaction level events
_SCSI	(value 8)	Send SCSI Transaction level events
_MCTP_MSG	(value 11)	Send MCTP_MSG Transaction level events
_MCTP_CMD	(value 12)	Send MCTP_CMD Transaction level events
_LM_CMD	(value 13)	Send LM Transaction level events
_LN	(value 14)	Send LN Transaction level events
_CXL_CM_LINKTRA	(value 17)	Send CXL.cache/mem Link Transaction level events
_CXL_DATA	(value 18)	Send CXL Data Transaction level events
_CXL_MEM_SPLITTRA	(value 19)	Send CXL.mem Split Transaction level events
_CXL_CACHE_SPLITTRA	(value 20)	Send CXL.cache Split Transaction level events

Example:

```
…
SendLevel( _PACKET); # Send packet level events
```

Remark:

If no level was specified, events of Packet level are sent to the script by default.

8.2 SendLevelOnly()

This function specifies that ONLY events of the specified transaction level should be sent to the script.

Format: SendLevelOnly(level)

Parameters:

Level	Value	Description
_PACKET	(value 0)	Send Packet level events
_LINK	(value 1)	Send Link Transaction level events
_PCIE_FLIT_LINK	(value 2)	Send PCIe Flit Link level events (TLPs)
_SPLIT	(value 3)	Send Split Transaction level events
_NVME	(value 4)	Send NVM Transaction level events
_NVMC	(value 10)	Send NVM Command level events
_AHCI	(value 6)	Send AHCI Transaction level events
_ATA	(value 7)	Send ATA Transaction level events
_MCTP_MSG	(value 12)	Send MCTP_MSG Transaction level events
_MCTP_CMD	(value 13)	Send MCTP_CMD Transaction level events
_LM_CMD	(value 14)	Send LM Transaction level events
_LN	(value 15)	Send LN Transaction level events
_CXL_CM_LINKTRA	(value 17)	Send CXL.cache/mem Link Transaction level events
_CXL_DATA	(value 18)	Send CXL Data Transaction level events
_CXL_MEM_SPLITTRA	(value 19)	Send CXL.mem Split Transaction level events
_CXL_CACHE_SPLITTRA	(value 20)	Send CXL.cache Split Transaction level events

Example:

```
…
SendLevelOnly( _PACKET ); # Send ONLY packet level events
```

8.3 DontSendLevel()

This function specifies that events of the specified transaction level should NOT be sent to the script.

Format: DontSendLevel(level)

Parameters:

Level	Value	Description
_PACKET	(value 0)	Do not send Packet level events
_LINK	(value 1)	Do not send Link Transaction level events
_SPLIT	(value 2)	Do not send Split Transaction level events
_NVME	(value 3)	Do not send NVM Transaction level events
_NVMC	(value 9)	Do not send NVM Command level events
_AHCI	(value 5)	Do not send AHCI Transaction level events
_ATA	(value 6)	Do not send ATA Transaction level events
_MCTP_MSG	(value 11)	Do not send MCTP_MSG Transaction level events
_MCTP_CMD	(value 12)	Do not send MCTP_CMD Transaction level events
_LM_CMD	(value 13)	Do not send LM Transaction level events
_LN	(value 14)	Do not send LN Transaction level events
_CXL_CM_LINKTRA	(value 17)	Do not send CXL.cache/mem Link Transaction level events
_CXL_DATA	(value 18)	Do not Send CXL Data Transaction level events
_CXL_MEM_SPLITTRA	(value 19)	Do not send CXL.mem Split Transaction level events
_CXL_CACHE_SPLITTRA	(value 20)	Do not send CXL.cache Split Transaction level events

Example:

```
…
DontSendLevel( _LINK); # DO NOT send link transaction level events
```

8.4 SendChannel()

This function specifies that events that have occurred on the specified channel should be sent to script.

Format: SendChannel(channel)

Parameters:

Channel	Value	Description
_CHANNEL_1	(= 1)	Send events from Upstream direction of the link (channel 1)
_CHANNEL_2	(= 2)	Send events from Downstream direction of the link (channel 2)

Example:

```
…
SendChannel(_CHANNEL_1); # Send events from Upstream direction of the link
```

8.5 SendChannelOnly()

This function specifies that ONLY events that have occurred on the specified channel should be sent to the script.

Format: SendChannelOnly(channel)

Parameters:

Channel	Value	Description
_CHANNEL_1	(= 1)	Send events from Upstream direction of the link (channel 1)
_CHANNEL_2	(= 2)	Send events from Downstream direction of the link (channel 2)

Example:

```
…
SendChannelOnly( _CHANNEL_1 ); # Send ONLY events from Upstream
                               # direction of the link
```

8.6 DontSendChannel ()

This function specifies that events that have occurred on the specified channel should NOT be sent to the script.

Format: DontSendChannel (channel)

Parameters:

Channel	Value	Description
_CHANNEL_1	(= 1)	DO NOT Send events from Upstream direction of the link (channel 1)
_CHANNEL_2	(= 2)	DO NOT Send events from Downstream direction of the link (channel 2)

Example:

```
…
DontSendChannel ( _CHANNEL_1 ); # DO NOT send events from Upstream
                                # direction of the link
```

8.7 SendAllChannels()

This function specifies that events that have occurred on ALL channels should be sent to the script.

Format: SendAllChannels ()

Example:

```
…
SendAllChannels (); # Send events from ALL channels
```

8.8 SendTraceEvent ()

This function specifies the events to be sent to the script.

Format: SendTraceEvent(event)

Parameters:

`event` Can have one of the following values:

Packet level events:

Event value	Description
_PKT_DLLP	Data Link Layer Packets (DLLP)
_PKT_TLP	Transaction Layer Packets (TLP)
_PKT_ORDERED_SET	Ordered Sets
_PKT_LINK_CONDITION	Link Conditions
_PKT_L0P_LINK_CONDITION	L0p Link Conditions
_PKT_SMBUS	System Management Bus (SMBus) packets
_PKT_PMUX	PMUX Packets
_PKT_EIE_EVENT	Electrical Idle Enter Packets
_PKT_INVALID	Invalid Packets
_PKT_PCIE_FLIT	PCIe Flit Records
_PKT_CXL_NULL_FLIT	CXL Null Flit Packets
_PKT_CXL_ALMP	CXL ALMP Packets
_PKT_CXL_CACHE_MEM	CXL Cache/Mem Packets
_PKT_CXL_IMPLIED_EDS	CXL Implied EDS Packets

Link Transaction level events:

Event value	Description
_LINK_MEMORY	Memory transactions
_LINK_IO	IO transactions
_LINK_CONFIG	Configuration transactions
_LINK_MESSAGE	Message transactions
_LINK_COMPLETION	Completion transactions

Split Transaction level events:

Event value	Description
_SPLIT_MEMORY	Memory transactions
_SPLIT_IO	IO transactions
_SPLIT_CONFIG	Configuration transactions

NVM Transaction level events:

Event value	Description
_NVME_CONTROLLER_REG	Transactions that read or write controller register
_NVME_DOORBELL_REG	Transactions that write doorbell register.
_NVME_ADMIN_SUBMISSION_CMD	Transactions that transfer admin submission command
_NVME_NVM_SUBMISSION_CMD	Transactions that transfer NVM submission command
_NVME_ADMIN_COMPLETION_CMD	Transactions that transfer admin completion command
_NVME_NVM_COMPLETION_CMD	Transactions that transfer NVM completion command
_NVME_IDX_DAT_REG	Transactions that target idx/dat registers (registers that are at fixed offset from BAR2)
_NVME_TRANSFERED_DATA	Transactions that target data which is referenced by PRP or SGL
_NVME_PRP	Generated for all PRP list transactions
_NVME_SGL	Generated for all SGL descriptors transactions

AHCI Transaction level events:

Event value	Description
_AHCI_HBA_REG	Transactions that use HBA registers
_AHCI_PORT_REG	Transactions that use Ports registers
_AHCI_CMND_LIST	Transactions that use Command list registers
_AHCI_RECEIVED_FIS	Transaction uses FIS registers
_AHCI_CMND_TABLE	Transaction uses Command table registers

MCTP Transaction level events:

Event value	Description
_MCTP_MSG_EVT_CONTROL	Messages used to support initialization and configuration of MCTP communication within an MCTP network
_MCTP_MSG_EVT_PLDM	Messages used to convey Platform Level Data Model (PLDM) traffic over MCTP
_MCTP_MSG_EVT_NC_SI_OVER_MCTP	Messages used to convey NC-SI Control traffic over MCTP
_MCTP_MSG_EVT_ETHERNET_OVER_MCTP	Messages used to convey Ethernet traffic over MCTP
_MCTP_MSG_EVT_NVME_OVER_MCTP	Messages used to convey NVMe (NVM Express) Management Messages over MCTP
_MCTP_MSG_EVT_VENDOR_DEFINED_PCI	Message type used to support VDMs where the vendor is identified using a PCI-based vendor ID
_MCTP_MSG_EVT_VENDOR_DEFINED_IANA	Message type used to support VDMs where the vendor is identified using an IANA-based vendor ID
_MCTP_MSG_EVT_RESERVED	Reserved

MCTP Command level events:

Event value	Description
_MCTP_CMD_EVT_CONTROL	Commands used to support initialization and configuration of MCTP communication within an MCTP network
_MCTP_CMD_EVT_PLDM	Commands used to convey Platform Level Data Model (PLDM) traffic over MCTP
_MCTP_CMD_EVT_NC_SI_OVER_MCTP	Commands used to convey NC-SI Control traffic over MCTP
_MCTP_CMD_EVT_NVME_OVER_MCTP	Commands used to convey NVMe (NVM Express) Management Messages over MCTP

Example:

```
…
SendTraceEvent( _PKT_TLP );
…

SendLevel( _LINK );
SendTraceEvent ( _LINK_MEMORY ); # Send memory Read and Write request
                                 # transactions to the script
```

8.9 DontSendTraceEvent()

This function specifies that the event specified in this function should not be sent to script.

Format: DontSendTraceEvent (event)

Parameters:

`event` See **SendTraceEvent()** for all possible values.

Example:

```
…
SendLevel( _LINK );                  # Send Link Transaction level events
SendTraceEvent ( _LINK_CONFIG );     # Send Configuration transactions
SendTraceEvent ( _LINK_COMPLETION ); # Send Completion transactions
SendTraceEvent ( _LINK_MESSAGE );    # Send Message transactions
…
if( SomeCondition )
{
DontSendTraceEvent (_LINK_CONFIG); # Don't send Cfg Request transactions
DontSendTraceEvent (_LINK_COMPLETION); # Don't send Completion transactions

# Only Message transactions are sent.
}
```

8.10 SendTraceEventOnly()

This function specifies that ONLY the event specified in this function is sent to the script.

Format: SendTraceEventOnly(event)

Parameters:

`event` See **SendTraceEvent()** for all possible values.

Remark: This function may be useful when many events are to be sent, but you need to send only one kind of event and turn off the rest.

Example:

```
…
SendLevel( _LINK );                    # Send Link Transaction level events
SendTraceEvent ( _LINK_CONFIG );       # Send Configuration  transactions
SendTraceEvent ( _LINK_COMPLETION  ); # Send Completion transactions
SendTraceEvent ( _LINK_MESSAGE  );     # Send Message transactions
…
if( SomeCondition )
{
      SendTraceEventOnly (_LINK_MEMORY );
      # Only Memory read/write request transactions are sent.
}
```

8.11 SendAllTraceEvents()

This function specifies that ALL trace events relevant for the selected transaction level are sent to the script.

Format: **SendAllTraceEvents ()**

Example:

```
…
SendLevel( _PACKET );    # Send packet level events
SendAllTraceEvents ( ); # All TLP, DLLP and Ordered Set packets
                         # are sent to the script
```

8.12 SendDllpType()

This function specifies more precise tuning (filtering in) for sending DLLP packets to the script.

Format: SendDllpType(dllp_type)

Parameters:

`dllp_type` Encoding of the DLLP type. This parameter may be one of the following values:

DLLP type values:

Constant	DLLP type
_DLLP_TYPE_ACK	Ack
_DLLP_TYPE_NAK	Nak
_DLLP_TYPE_INIT_FC1_P	InitFC1-P
_DLLP_TYPE_INIT_FC1_NP	InitFC1-NP
_DLLP_TYPE_INIT_FC1_CPL	InitFC1-Cpl
_DLLP_TYPE_INIT_FC2_P	InitFC2-P
_DLLP_TYPE_INIT_FC2_NP	InitFC2-NP
_DLLP_TYPE_INIT_FC2_CPL	InitFC2-Cpl
_DLLP_TYPE_UPDATE_FC_P	UpdateFC-P
_DLLP_TYPE_UPDATE_FC_NP	UpdateFC-NP
_DLLP_TYPE_UPDATE_FC_CPL	UpdateFC-Cpl
_DLLP_TYPE_NOP	NOP
_DLLP_TYPE_PM	All Power Management DLLP types
_DLLP_TYPE_INVALID	Invalid DLLP types
_DLLP_TYPE_INIT_FC	All InitFC DLLP types
_DLLP_TYPE_UPDATE_FC	All UpdateFC DLLP types
_DLLP_TYPE_DATA_LINK_FEATURE	Data Link Feature
_DLLP_TYPE_NOP2	NOP2 (PCIe Flit Mode)
_DLLP_TYPE_LINK_MANAGEMENT	Link Management (PCIe Flit Mode)
_DLLP_TYPE_MR	All MR DLLP types
_ANY_TYPE	All possible DLLP types

Example:

```
SendDllpType(_DLLP_TYPE_ACK );      # Send Ack DLLPs to the script
…

SendDllpType(_DLLP_TYPE_UPDATE_FC); # Send all UpdateFC DLLPs
```

8.13 FilterDllpType()

This function specifies more precise tuning (filtering out) for sending DLLP packets to the script.

Format: FilterDllpType(dllp_type)

Parameters:

`dllp_type`	Encoding of the DLLP type. This parameter may be one of the values defined for the **SendDllpType()** function.

Example:

```
SendDllpType(_DLLP_TYPE_INIT_FC);       # Send all InitFC DLLPs to the script
FilterDllpType(_DLLP_TYPE_INIT_FC1_CPL); # Don't send InitFCs for Completions
FilterDllpType(_DLLP_TYPE_INIT_FC2_CPL);

# Only InitFC DLLPs for Posted and Non-posted requests are sent to the script
```

8.14 SendTlpType()

This function specifies more precise tuning (filtering in) for sending TLP packets to the script.

Format: SendTlpType(tlp_type)

Parameters:

`tlp_type` Encoding of the TLP type. This parameter may be one of the following values:

TLP type values:

Constant	TLP type
_TLP_TYPE_INVALID	Invalid TLP types
_TLP_TYPE_MRD32	Memory Read Request, 32-bit address format
_TLP_TYPE_MRDLK32	Memory Read Request - Locked, 32-bit address format
_TLP_TYPE_MWR32	Memory Write Request, 32-bit address format
_TLP_TYPE_MRD64	Memory Read Request, 64-bit address format
_TLP_TYPE_MRDLK64	Memory Read Request – Locked, 64-bit address format
_TLP_TYPE_MWR64	Memory Write Request, 64-bit address format
_TLP_TYPE_IORD	I/O Read Request
_TLP_TYPE_IOWR	I/O Write Request
_TLP_TYPE_CFGRD_0	Configuration Read Type 0
_TLP_TYPE_CFGWR_0	Configuration Write Type 0
_TLP_TYPE_CFGRD_1	Configuration Read Type 1
_TLP_TYPE_CFGWR_1	Configuration Write Type 1
_TLP_TYPE_MSG	Message Request
_TLP_TYPE_MSGD	Message Request with Data payload
_TLP_TYPE_MSGAS	Message for Advanced Switching
_TLP_TYPE_MSGASD	Message for Advanced Switching with Data
_TLP_TYPE_CPL	Completion
_TLP_TYPE_CPLD	Completion with Data
_TLP_TYPE_CPLLK	Completion for Locked Memory Read
_TLP_TYPE_CPLDLK	Completion for Locked Memory Read with Data
_TLP_TYPE_ID_FETCHADD32	Fetch and Add AtomicOp Request, 32-bit address format
_TLP_TYPE_ID_FETCHADD64	Fetch and Add AtomicOp Request, 64-bit address format
_TLP_TYPE_ID_SWAP32	Unconditional Swap AtomicOp Request, 32-bit address format
_TLP_TYPE_ID_SWAP64	Unconditional Swap AtomicOp Request, 64-bit address format
_TLP_TYPE_ID_CAS32	Compare and Swap AtomicOp Request, 32-bit address format
_TLP_TYPE_ID_CAS64	Compare and Swap AtomicOp Request, 64-bit address format
_TLP_TYPE_ID_DMWR32	Deferrable Memory Write Request, 32-bit address format
_TLP_TYPE_ID_DMWR64	Deferrable Memory Write Request, 64-bit address format
_TLP_TYPE_MEMORY	All Memory Request TLP types
_TLP_TYPE_IO	All I/O Request TLP types
_TLP_TYPE_CONFIG	All Configuration Request TLP types
_TLP_TYPE_MESSAGE	All Message Request TLP types

Constant	TLP type
_TLP_TYPE_COMPLETION	All Completion TLP types
_ANY_TYPE	All possible TLP types
_PE_ALL_TYPES	All possible TLP types

Example:

```
SendTlpType( _TLP_TYPE_ID_MSG ); # Send Message Request TLPs to the script
…
SendTlpType( _TLP_TYPE_MEMORY ); # Send all Memory Request TLPs to the script
```

8.15 FilterTlpType()

This function specifies more precise tuning (filtering out) for sending TLP packets to the script.

Format: FilterTlpType(tlp_type)

Parameters:

`tlp_type` Encoding of the TLP type.
This parameter may be one of the values defined for the **SendTlpType()** function.

Example:

```
SendTlpType(_TLP_TYPE_CONFIG); # Send all Configuration Request TLPs
                               # to the script
FilterTlpType(_TLP_TYPE_ID_CFGRD_1); # Don't send Type 1 requests
FilterTlpType(_TLP_TYPE_ID_CFGWR_1);

# Only Type 0 Configuration Request TLPs are sent to the script
```

8.16 SendPCIeFlitType()

Format: SendPCIeFlitType(flit_type)

Parameters:

`flit_type` Encoding of the PCIe Flit type. This parameter may be one of the following values:

PCIe Flit type values:

Constant	PCIe Flit type
_PCIE_FLIT_TYPE_IDLE	IDLE
_PCIE_FLIT_TYPE_NOP	NOP
_PCIE_FLIT_TYPE_PAYLOAD	Payload
_ANY_TYPE	All possible PCIe Flit types
_PE_ALL_TYPES	All possible PCIe Flit types

Example:

```
FilterPCIeFlitType( _ANY_TYPE ); # Reset filtering mask
SendPCIeFlitType( _PCIE_FLIT_TYPE_IDLE ); # Send only IDLE Flits to the script
```

8.17 FilterPCIeFlitType()

This function specifies more precise tuning (filtering out) for sending PCIe Flit packets to the script.

Format: FilterPCIeFlitType(flit_type)

Parameters:

`flit_type` Encoding of the PCIe Flit type.
This parameter may be one of the values defined for the **SendPCIeFlitType()** function.

Example:

```
SendTraceEvent( _PKT_PCIE_FLIT ); # Send all PCIe Flit types
FilterPCIeFlitType(_PCIE_FLIT_TYPE_NOP); # Don't send NOP Flits
```

8.18 SendOrderedSetType()

This function specifies more precise tuning (filtering in) for sending Ordered Set packets to the script.

Format: SendOrderedSetType(set_type)

Parameters:

`set_type` Encoding of the Ordered Set type. This parameter may be one of the following values:

Ordered Set type values:

Constant	OS type
_ORDSET_TYPE_TS0	Training Sequence Type 0 (Gen6 only)
_ORDSET_TYPE_TS1	Training Sequence Type 1
_ORDSET_TYPE_TS2	Training Sequence Type 2
_ORDSET_TYPE_TS1_MOD	Modified Training Sequence Type 1
_ORDSET_TYPE_TS2_MOD	Modified Training Sequence Type 2
_ORDSET_TYPE_FTS	Fast Training Sequence
_ORDSET_TYPE_EIOS	EIOS
_ORDSET_TYPE_SKIP	Skip
_ORDSET_TYPE_PATN	Pattern
_ORDSET_TYPE_PATN_MODIFIED	Pattern Modified (Gen1/2 only)
_ORDSET_TYPE_EIEOS	EIEOS
_ORDSET_TYPE_SDS	SDS
_ANY_TYPE	All possible Ordered Set types

Example:

```
SendOrderedSetType(_ORDSET_TYPE_FTS); # Send Fast Training Sequences
                                      # to the script
…
```

8.19 FilterOrderedSetType()

This function specifies more precise tuning (filtering out) for sending Ordered Set packets to the script.

Format: FilterOrderedSetType(set_type)

Parameters:

`set_type`	Encoding of the Ordered Set type. This parameter may be one of the values defined for the **SendOrderedSetType()** function.

Example:

```
FilterOrderedSetType(_ORDSET_TYPE_SKIP); # Don't send Skip packets.
```

8.20 EnableNvmCommandImplicitStartTime()

This function changes behavior for timestamp of NVM Command. For implicitly started commands it will be taken from the first implicit entry rather than from the first sub-transaction.

Format: EnableNvmCommandImplicitStartTime()

8.21 DisableNvmCommandImplicitStartTime()

This function restores default behavior for timestamp of NVM Command. For implicitly started commands it will be taken from the first sub-transaction.

Format: DisableNvmCommandImplicitStartTime()

8.22 SendLtssm()

This function specifies that Ltssm events should be sent to the script.

Format: SendLtssm () or SendLtssm (channels_set)

Parameters:

`channels_set` Specify which ltssm channels should be send to VSE. For traces with jammed traffic function takes parameter, it may be one of the following values:

Constant	LTSSM type
LTSSM_AJ_CHANNELS	Send only pre –jammed traffic data
LTSSM_JA_CHANNELS	Send only post –jammed traffic data
LTSSM_ALL_CHANNELS	Send all pre- and post-jammed data

Example:

```
…
SendLtssm(); # Send Ltssm events
```

8.23 SetLtssmIgnoreProperties()

This function specifies that Ltssm specified packets errors should be filtered.

Format: SetLtssmIgnoreProperties (ignore_ts, ignore_fts, ignore_eieos, ignore_dllp, ignore_tlp)

Parameters:

Function	Description
ignore_ts	Specify if ts packets errors should be filtered
ignore_fts	Specify if fts packet errors should be filtered
ignore_eieos	Specify if eieos packet errors should be filtered
ignore_dllp	Specify if dllp packet errors should be filtered
ignore_tlp	Specify if tlp packet errors should be filtered

NOTE: Each of the parameters may be one of the following values: 1(true), 0(false).

Example:

```
…
SetLtssmIgnoreProperties(1, 1, 1, 1, 0);
```

8.24 FilterJammerActionType()

This function specifies more precise tuning (filtering out) for sending jammer action packets to the script.

Format: **FilterJammerActionType (action_type)**

Parameters:

`action type` jammer action type. This parameter may be one of the following values:

Constant	Jammer action type
_JAMMER_DELETE	Delete
_JAMMER_REPLACE	Replace
_JAMMER_MODIFY	Modify
_JAMMER_INS_PKT_X_AFTER	Insert packet
_JAMMER_SIDEBAND	Sideband signal
_JAMMER_INS_ERROR	Insert error

8.25 SetNFMLtssmIgnorePropertiesEx()

This function specifies that Ltssm specified packets errors should be filtered.

Format: SetNFMLtssmIgnorePropertiesEx (ignore_errors)

Non Flit bitfield:

Name	Bit	Value
IgnoreTS	1	0x1
IgnoreFTS	2	0x2
IgnoreEIEOS	3	0x4
IgnoreDLLP	4	0x8
IgnoreTLP	5	0x10
IgnoreEIOS	6	0x20

Example:

```
ignore_errors = 0x3F;
SetNFMLtssmIgnorePropertiesEx(ignore_errors);
```

8.26 GetNFMLtssmIgnorePropertiesEx()

This function specifies that Ltssm specified packets errors should be filtered.

Format: **GetNFMLtssmIgnorePropertiesEx (ignore_errors)**

Non Flit bitfield:

Name	Bit	Value
IgnoreTS	1	0x1
IgnoreFTS	2	0x2
IgnoreEIEOS	3	0x4
IgnoreDLLP	4	0x8
IgnoreTLP	5	0x10
IgnoreEIOS	6	0x20

Example:

```
ignore_errors = GetNFMLtssmIgnorePropertiesEx();
```

8.27 SetFMLtssmIgnorePropertiesEx()

This function specifies that Ltssm specified packets errors should be filtered.

Format: **SetFMLtssmIgnorePropertiesEx (ignore_errors)**

Flit bitfield:

Name	Bit	Value
IgnoreTS	1	0x1
IgnoreEIEOS	2	0x2
IgnoreFlit	3	0x4
IgnoreEIOS	4	0x8

Example:

```
ignore_errors = 0xF;
SetFMLtssmIgnorePropertiesEx(ignore_errors);
```

8.28 GetFMLtssmIgnorePropertiesEx()

This function specifies that Ltssm specified packets errors should be filtered.

Format: **GetFMLtssmIgnorePropertiesEx()**

Flit bitfield:

Name	Bit	Value
IgnoreTS	1	0x1
IgnoreEIEOS	2	0x2
IgnoreFlit	3	0x4
IgnoreEIOS	4	0x8

Example:

```
ignore_errors = GetFMLtssmIgnorePropertiesEx();
```

9 Timer Functions

This group of functions covers VSE capability to work with timers –--- internal routines that repeatedly measure a timing interval between different events.

9.1 VSE Time Object

A VSE time object is a special object that presents time intervals in verification scripts.
From point of view of the **CSL**, the verification script time object is a “list”-object of two or three elements.
(Please see the *CSL Manual* for more details about CSL types.)

[seconds, nanoseconds, (picoseconds)]

Note: The best way to construct a VSE time object is to use the **Time()**/**Time2()** function (see below).

9.2 SetTimer()

Starts the timing calculation from the event where this function was called.

Format: SetTimer(timer_id = 0)

Parameters:

`timer_id` Unique timer identifier

Example:

```
SetTimer();   # Start timing for timer with id = 0.
SetTimer(23); # Start timing for timer with id = 23.
```

Remark:

If this function is called a second time for the same timer ID, it resets the timer and starts timing calculations again from the point where it was called.

9.3 KillTimer()

Stops timing calculation for a specific timer and frees related resources.

Format: KillTimer(timer_id = 0)

Parameters:

`timer_id` Unique timer identifier

Example:

```
KillTimer();   # Stop timing for timer with id = 0.
KillTimer(23); # Stop timing for timer with id = 23.
```

9.4 GetTimerTime()

Retrieve the timing interval from the specific timer.

Format: GetTimerTime (timer_id = 0)

Parameters:

`timer_id` Unique timer identifier

Return values:

Returns VSE time object from timer with id = timer_id.

Example:

```
GetTimerTime ();   # Retrieve timing interval for timer with id = 0.
GetTimerTime (23); # Retrieve timing interval for timer with id = 23.
```

Remark :

This function, when called, does not reset the timer.

10 Time Construction Functions

This group of functions are used to construct VSE time objects.

10.1 Time()

Constructs a verification script time object.

Format: **Time(nanoseconds)**
Time(seconds, nanoseconds)

Return values:

First function returns **[0, nanoseconds]**
Second function returns **[seconds, nanoseconds]**

Parameters:

`nanoseconds`	Number of nanoseconds in specified time
`seconds`	Number of seconds in specified time

Example:

```
Time ( 50 * 1000 ); # - create time object of 50 microseconds
Time (3, 100);      # - create time object of 3 seconds and 100 nanoseconds
Time( 3 * MICRO_SECS );    # - create time object of 3 microseconds
Time( 4 * MILLI_SECS    ); # - create time object of 4 milliseconds
```

Note: MICRO_SECS and **MILLI_SECS** are constants defined in **VS_constants.inc.**

10.2 Time2()

Constructs a verification script time object.

Format: **Time2(nanoseconds)**
Time2(seconds, nanoseconds)
Time2(seconds, nanoseconds, picoseconds)

Return values:

First function returns **[0, nanoseconds]**
Second function returns **[seconds, nanoseconds]**
Third function returns **[seconds, nanoseconds, picoseconds]**

Parameters:

`nanoseconds`	Number of nanoseconds in specified time
`seconds`	Number of seconds in specified time
`picoseconds`	Number of picoseconds in specified time

Example:

```
      Time2 ( 50 * 1000 ); # - create time object of 50 microseconds
      Time2 (3, 100);      # - create time object of 3 seconds and 100 nanoseconds
      Time2 (2, 250, 70);  # - create time object of 2 seconds, 250 nanoseconds and
70 picoseconds
      Time2( 3 * MICRO_SECS );    # - create time object of 3 microseconds
      Time2( 4 * MILLI_SECS    ); # - create time object of 4 milliseconds
```

Note: MICRO_SECS and **MILLI_SECS** are constants defined in **VS_constants.inc.**

11 Time Calculation Functions

This group of functions covers VSE capability to work with “time” – VSE time objects.

11.1 AddTime()

Adds two VSE time objects

Format: AddTime(time1, time2)

Return values:

Returns VSE time object representing the time interval equal to the sum of **time_1** and **time_2.**

Parameters:

`time_1`	VSE time object representing the first time interval
`time_2`	VSE time object representing the second time interval

Example:

```
t1 = Time(100);
t2 = Time(2, 200);
t3 = AddTime( t1, t2 ) # Returns VSE time object = 2 sec 300 ns.
```

11.2 SubtractTime()

Subtract two VSE time objects

Format: SubtractTime (time1, time2)

Return values:

Returns VSE time object representing the time interval equal to the difference between **time_1** and **time_2.**

Parameters:

`time_1`	VSE time object representing the first time interval
`time_2`	VSE time object representing the second time interval

Example:

```
t1 = Time(100);
t2 = Time(2, 200);
t3 = SubtractTime ( t2, t1 ) # Returns VSE time object = 2 sec 100 ns.
```

11.3 MulTimeByInt()

Multiplies VSE time object by integer value

Format: MulTimeByInt (time, mult)

Return values:

Returns VSE time object representing the time interval equal to the product of **time * mult.**

Parameters:

`time`	VSE time object
`mult`	multiplier, integer value

Example:

```
t  = Time(2, 200);
t1 = MulTimeByInt ( t, 2 ) # Returns VSE time object = 4 sec 400 ns.
```

11.4 DivTimeByInt()

Divides VSE time object by integer value

Format: DivTimeByInt (time, div)

Return values:

Returns VSE time object representing the time interval equal to the quotient of **time / div.**

Parameters:

`time`	VSE time object
`div`	divisor, integer value

Example:

```
t  = Time(2, 200);
t1 = DivTimeByInt ( t, 2 ) # Returns VSE time object = 1 sec 100 ns.
```

12 Time Logical Functions

This group of functions covers VSE capability to compare VSE time objects

12.1 IsEqualTime()

Verifies that one VSE time object is equal to the other VSE time object.

Format: IsEqualTime (time1, time2)

Return values:

Returns 1 if **time_1** is equal to **time_2**, returns 0 otherwise.

Parameters:

`time_1`	VSE time object representing the first time interval
`time_2`	VSE time object representing the second time interval

Example:

```
t1 = Time(100);
t2 = Time(500);
If( IsEqualTime( t1, t2 ) ) DoSomething();
```

12.2 IsLessTime()

Verifies that one VSE time object is less than the other VSE time object

Format: IsLessTime (time1, time2)

Return values:

Returns 1 if **time_1** is less than **time_2**, returns 0 otherwise.

Parameters:

`time_1`	VSE time object representing the first time interval
`time_2`	VSE time object representing the second time interval

Example:

```
t1 = Time(100);
t2 = Time(500);
If( IsLessTime ( t1, t2 ) ) DoSomething();
```

12.3 IsGreaterTime()

Verifies that one VSE time object is greater than the other VSE time object

Format: IsGreaterTime (time1, time2)

Return values:

Returns 1 if **time_1** is greater than **time_2**, returns 0 otherwise.

Parameters:

`time_1` VSE time object representing the first time interval
`time_2` VSE time object representing the second time interval

Example:

```
t1 = Time(100);
t2 = Time(500);
If( IsGreaterTime ( t1, t2 ) ) DoSomething();
```

12.4 IsTimeInInterval()

Verifies that a VSE time object is greater than some VSE time object and less than the other VSE time object.

Format: IsTimeInInterval(min_time, time, max_time)

Return values:

Returns 1 if **min_time <= time <= max_time**, returns 0 otherwise.

Parameters:

`time_1`	VSE time object representing the first time interval
`time_2`	VSE time object representing the second time interval

Example:

```
t1 = Time(100);
t  = Time(400);
t2 = Time(500);
If( IsTimeInInterval ( t1, t, t2 ) ) DoSomething();
```

13 Time Text Functions

This group of functions covers VSE capability to convert VSE time objects into text strings.

13.1 TimeToText()

Converts a VSE time object into text.

Format: TimeToText (time)

Return values:

Returns a text representation of VSE time object

Parameters:

`time` VSE time object

Example:

```
t = Time(100);
ReportText( TimeToText( t ) ); # See below details for ReportText() function
```

14 Output Functions

This group of functions covers VSE capability to present information in the output window.

14.1 ReportText()

Outputs text in the output window related to the verification script.

Format: ReportText (text)

Parameters:

`text`	Text variable, constant, or literal

Example:

```
…
ReportText ( “Some text” );
…
t = “Some text”
ReportText ( t );
…
num_of_frames = in.NumOfFrames;
text = Format( “Number of frames : %d”, num_of_frames );
ReportText ( text );
…
x = 0xAAAA;
y = 0xBBBB;
text = FormatEx( “x = 0x%04X, y = 0x%04X”, x, y );
ReportText( “Text : ” + text );
…
```

14.2 EnableOutput()

Enables showing information in the output window and sending COM reporting notifications to COM clients.

Format: **EnableOutput ()**

Example:

```
EnableOutput ( );
```

14.3 DisableOutput()

Disables showing information in the output window and sending COM reporting notifications to COM clients.

Format: DisableOutput ()

Example:

```
DisableOutput ();
```

15 Information Functions

15.1 GetTraceName()

This function returns the filename of the trace file being processed by VSE.
If the script is being run over a multi-segmented trace, this function returns the path to the segment being processed.

Format: GetTraceName(filepath_compatible)

Parameters:

filepath_compatible	If this parameter is present and not equal to 0, the returned value may be used as part of the filename.

Example:

```
ReportText( "Trace name : " + GetTraceName() );
…
File = OpenFile( "C:\\My Files\\" + GetTraceName(1) + "_log.log" );

# For trace file with path - D:\Some Traces\Data.pex
# GetTraceName(1) returns - "D_Some Traces_Data.pex"
```

15.2 GetScriptName()

This function returns the name of the verification script where this function is called.

Format: GetScriptName()

Example:

```
ReportText( "Current script : " + GetScriptName() );
```

15.3 GetApplicationFolder()

This function returns the full path of the folder where the PCIe Protocol Suite™ application was started.

Format: GetApplicationFolder()

Example:

```
ReportText( "PCIe Protocol Suite folder : " + GetApplicationFolder () );
```

15.4 GetCurrentTime()

This function returns the string representation of the current system time.

Format: GetCurrentTime()

Example:

```
ReportText( GetCurrentTime() ); # Yields "February 10, 2004, 5:49 PM"
```

15.5 GetEventSegNumber()

In case if a multi-segmented trace is being processed, this function returns the index of the segment for the current event.

Note: When a multi-segmented trace file (extension ***.pem**) is processed by VSE, different trace events in different segments of the same trace file may have the same indexes (value stored in **in.Index** input context members), but they have different segment numbers.

Format: GetEventSegNumber()

Example:

```
ReportText( Format( "Current segment = %d", GetEventSegNumber() ) );
```

15.6 GetTriggerPacketNumber()

This function returns the number of the trigger packet in the trace. In case no trigger event was recorded in the trace, a value of 0xFFFFFFFF is returned.

Format: **GetTriggerPacketNumber()**

Example:

```
ReportText( FormatEx( “Trigger packet # : %i”, GetTriggerPacketNumber() );
```

15.7 TraHasError ()

This function returns non-zero value if transaction has passed error.

Format: TraHasError(error_code)

Example:

```
if ( TraHasError( _NVMC_ERROR_INCOMPLETE_SUB_TRA ) )
{
        ReportText("Incompete sub transaction") );
}
```

15.8 GetSoftwareVersion ()

This function returns software version.

Format: GetSoftwareVersion ( )

Example:

```
ReportText( "Current software version : " + GetSoftwareVersion() );
```

15.9 GetScriptFullPath ()

This function returns full path to the file with the name and extension.

Format: GetScriptFullPath ( )

Example:

```
ReportText( "Current script full path : " + GetScriptFullPath() );
```

16 Navigation Functions

16.1 GotoEvent()

This function forces the application to jump to some trace event and show it in the main trace view.

Format: **GotoEvent(level, index, segment)**
GotoEvent()

Parameters:

`level` Transaction level of the event to jump to (possible values: _PACKET, _LINK, _SPLIT, _NVME, _NVMC, _AHCI, _ATA, _MCTP_MSG, _MCTP_CMD, _LM_CMD, _LN)
`index` Transaction index of the event to jump to
`segment` Segment index of the event to jump to.
If omitted, the current segment index is used.

Remarks:

If no parameters were specified, the application jumps to the current event being processed by VSE. The **segment** parameter is used only when the verification script is running over a multi-segmented trace (extension: ***.pem**). For regular traces it is ignored.

If wrong parameters were specified (like an index exceeding the maximum index for the specified transaction level), the function does nothing and an error message is sent to the output window.

Example:

```
…
if( Something == interesting ) GotoEvent(); # go to the current event
…
if( SomeCondition )
{
 interesting_segment = GetEventSegNumber();
 interesting_level = in.Level;
 interesting_index = in.Index;
}
…
OnFinishScript()
{
     …
      # go to the interesting event…
      GotoEvent( interesting_level, interesting_index, interesting_segment );
}
```

16.2 SetMarker()

This function sets a marker for some trace event.

Format: **SetMarker(marker_text)**
SetMarker(marker_text, level, index, segment)

Parameters:

`marker_text`	Text of the marker
`level`	Transaction level of the event to jump to (possible values: _PACKET, _LINK, _SPLIT, _NVME, _NVMC, _AHCI, _ATA, _MCTP_MSG, _MCTP_CMD, _LM_CMD, _LN)
`index`	Transaction index of the event to jump to
`segment`	Segment index of the event to jump to. If omitted, the current segment index is used.

Remarks:

If no parameters were specified, other than **marker_text,** the application sets a marker to the current event being processed by VSE. The **segment** parameter is used only when a verification script is running over a multi-segmented trace (extension: ***.pem**). For regular traces it is ignored.

If wrong parameters were specified (like an index exceeding the maximum index for a specified transaction level), the function does nothing and an error message is sent to the output window.

Example:

```
…
# set marker to the current event
if( Something == interesting ) SetMarker( "!!! Something cool !!!" );
…
if( SomeCondition )
{
 interesting_segment = GetEventSegNumber();
 interesting_level = in.Level;
 interesting_index = in.Index;
}
…
OnFinishScript()
{
     …
      # set marker to the interesting event…
      SetMarker(  "  !!! Cool Marker !!! ", interesting_level,
                                            interesting_index,
                                            interesting_segment );

      # go to the interesting event…
      GotoEvent( interesting_level, interesting_index, interesting_segment );
}
```

17 File Functions

This group of functions covers VSE capabilities to work with the external files.

17.1 OpenFile()

This function opens a file for writing.

Format: OpenFile(file_path, append)

Parameters:

`file_path`	Full path to the file to open. (For '\' use '\\')
`append`	This parameter (if present and not equal to 0) specifies that VSE should append to the contents of the file. Otherwise, the contents of the file are overwritten.

Return Values:

The "handle" to the file to be used in other file functions.

Example:

```
…
set file_handle = 0;
…
file_handle = OpenFile( "D:\\Log.txt" ); # Opens file, the previous contents
                                         # are erased.
…
WriteString( file_handle, "Some Text1" );  # Write text string to file
WriteString( file_handle, "Some Text2" );  # Write text string to file
…
CloseFile( file_handle ); # Closes file
…
# Opens file, the following file operations append to contents of the file.
file_handle = OpenFile( GetApplicationFolder() + "Log.txt", _APPEND );
```

17.2 CloseFile()

This function closes an opened file.

Format: CloseFile(file_handle)

Parameters:

`file_handle` File “handle”

Example:

```
…
set file_handle = 0;
…
file_handle = OpenFile( “D:\\Log.txt” ); # opens file, the previous contents are
                                         # erased.
…
WriteString( file_handle, “Some Text1” );  # write text string to file
WriteString( file_handle, “Some Text2” );  # write text string to file
…
CloseFile( file_handle ); # closes file
…
```

17.3 WriteString()

This function writes a text string to the file.

Format: WriteString(file_handle, text_string)

Parameters:

`file_handle`	File “handle”
`text_string`	Text string”

Example:

```
…
set file_handle = 0;
…
file_handle = OpenFile( “D:\\Log.txt” );  # Opens file, the previous contents
                                          # are erased.
…
WriteString( file_handle, “Some Text1” ); # Write text string to file
WriteString( file_handle, “Some Text2” ); # Write text string to file
…
CloseFile( file_handle );                 # Closes file
…
```

17.4 Write()

This function writes data to the file.

Format: Write(file_handle, value, num_of_bytes)

Parameters:

file_handle	"handle" to the file previously opened by OpenFile()
value	Data to write
num_of_bytes	Optional parameter specifying how many bytes to take from the value parameter (if omitted, length is calculated automatically based on the value type)

Remarks:

This function is primarily for binary files. It can be used for text files only if the value parameter is a string of text. In that case, it is equivalent to the WriteString() function and the num_of_bytes parameter is ignored.

Example:

```
…
set BinFile = 0;
…
BinFile = OpenFile("C:\\data.bin", 0, _FO_BINARY);
…
```

Write a string to the binary file.

```
Write(BinFile, "All we need is love!!!");
```

Write a substring (“All”) to the binary file.

```
Write(BinFile, "All we need is love!!!", 3);
```

Write Integer or WORD to the binary file:

```
val = 0xBEEF;
Write(BinFile, val); # Writes integer = (EF BE 00 00) to the binary file.
Write(BinFile, val, 2); # Writes WORD = (EF BE) to the binary file.
```

Write a byte chain to the binary file.

```
Write(BinFile, 'AABBCCDDEEFF12345678');
```

Write a list of values to the binary file.

```
Write(BinFile, [ 0xAA, "USB", 12, 0xBEEF ]);
…
```

```
CloseFile(BinFile); # Closes the binary file.
```

17.5 ShowInBrowser()

This function allows you to open a file in the Windows® Explorer. If the extension of the file has the application registered to open files with such extensions, it is launched. For instance, if Internet Explorer is registered to open files with extensions ***.htm** and the file handle passed to **ShowInBrowser()** function belongs to a file with such an extension,this file is opened in the Internet Explorer.

Format: ShowInBrowser (file_handle)

Parameters:

`file_handle` File "handle"

Example:

```
…
set html_file = 0;
…
html_file = OpenFile( "D:\\Log.htm" );
…
WriteString( html_file, "<html><head><title>LOG</title></head>" );
WriteString( html_file, "<body>" );
…
WriteString( html_file, "</body></html>" );
ShowInBrowser( html_file ); # opens the file in Internet Explorer
CloseFile( html_file );
…
```

17.6 OpenFileEx()

VSE should open a file in read or write mode based on Boolean input parameter value.

Format: OpenFile(file_path, read, append)

Parameters:

`file_path`	Full path to the file to open. (For '\' use '\\')
`read`	This parameter specifies that VSE should open file in read mode or in wrire mode.
`append`	This parameter (if present and not equal to 0) specifies that VSE should append to the contents of the file. Otherwise, the contents of the file are overwritten.
`text_mode`	This parameter (if present and not equal to True) specifies that file should be open in binary mode. Otherwise, file should be opened in text mode.

Return Values:

The "handle" to the file to be used in other file functions.

Example:

```
…
set file_handle = 0;
set readMode = 0;
…
```

```
      file_handle = OpenFileEx( "D:\\Log.txt", readMode); # Opens file, the previous
contents
                                                  # are erased.
      …
      WriteString( file_handle, "Some Text1" );  # Write text string to file
      WriteString( file_handle, "Some Text2" );  # Write text string to file
      …
      CloseFile( file_handle ); # Closes file
      …
      # Opens file, the following file operations append to contents of the file.
      set readMode = 1;
      file_handle = OpenFileEx( GetApplicationFolder() + "Log.txt", readMode,
_APPEND );


      …
```

17.7 ReadDword()

This function reads a file

Format: ReadDword(file_handle)

Parameters:

`file_handle` The "handle" to the file to be used in other file functions

Return Values:

The content of buffer file_handle points to .

Example:

```
…
readMode = 1;

file_handle = OpenFileEx( file_path, readMode );
if( file_handle != 0 )
{
      register_data = ReadDword( file_handle );
      reg_status = register_data >> 26;
      …
      CloseFile( file_handle );
}
```

18 COM/Automation Communication Functions

This group of functions covers VSE capabilities to communicate with COM/Automation clients connected to the PCIe Protocol Suite™ application. (Please refer to the *PCIe Protocol Suite Automation Manual* for the details on how to connect to the PCIe Protocol Suite application and VSE)

18.1 NotifyClient()

This function allows you to send information to COM/Automation client applications in a custom format. The client application receives a VARIANT object, which it is supposed to parse.

Format: NotifyClient(param_list)

Parameters:

`param_list` List of parameters to be sent to the client application. Each parameter might be an integer, string or list.
(See *CSL Manual* for details about data types available in CSL.)

Because the list itself may contain integers, strings, or other lists – it is possible to send complicated messages.
(Lists should be treated as arrays of VARIANTs.)

Example:

```
…
if( SomeCondition() )
{
      NotifyClient( 2, [ in.Index, in.Level, "CHANNEL 2", "TLP",
                                    TimeToText( in.Time )] );
}
…
# Here we sent 2 parameters to clients applications :
#  2 ( integer ),
# [ in.Index, in.Level, "CHANNEL 2", "TLP", TimeToText( in.Time )] ( list )
```

Remark:

See an example of handling this notification by client applications and parsing code in the *PE Automation* document.

19 User Input Functions

19.1 MsgBox()

Displays a message in a dialog box, waits for the user to click a button, and returns an Integer indicating which button the user clicked.

Format: MsgBox(prompt, type, title)

Parameters:

`prompt` Required. String expression displayed as the message in the dialog box.

`type` Optional. Numeric expression that is the sum of values specifying the number and type of buttons to display, the icon style to use, the identity of the default button, and the modality of the message box. If omitted, the default value for buttons is **_MB_OK**. (See the list of possible values in the table below)

`title` Optional. String expression displayed in the title bar of the dialog box. If you omit the title, the script name is placed in the title bar.

The **type** argument values are:

Constant	Description
_MB_OKONLY	Display **OK** button only (by Default).
_MB_OKCANCEL	Display **OK** and **Cancel** buttons.
_MB_RETRYCANCEL	Display **Retry** and **Cancel** buttons.
_MB_YESNO	Display **Yes** and **No** buttons.
_MB_YESNOCANCEL	Display **Yes**, **No**, and **Cancel** buttons.
_MB_ABORTRETRYIGNORE	Display **Abort**, **Retry**, and **Ignore** buttons.
_MB_EXCLAMATION	Display **Warning Message** icon.
_MB_INFORMATION	Display **Information Message** icon.
_MB_QUESTION	Display **Warning Query** icon.
_MB_STOP	Display **Critical Message** icon.
_MB_DEFBUTTON1	First button is default.
_MB_DEFBUTTON2	Second button is default.
_MB_DEFBUTTON3	Third button is default.
_MB_DEFBUTTON4	Fourth button is default.

Return Values:

This function returns an integer value indicating which button the user clicked.

Constant	Description
_MB_OK	**OK** button was clicked.
_MB_CANCEL	**Cancel** button was clicked.
_MB_YES	**Yes** button was clicked.
_MB_NO	**No** button was clicked.
_MB_RETRY	**Retry** button was clicked.
_MB_IGNORE	**Ignore** button was clicked.
_MB_ABORT	**Abort** button was clicked.

Remark:

This function works only for VS Engines controlled via the GUI. For VSEs controlled by COM/Automation clients, it does nothing.

This function "locks" the PCIe Protocol Suite™ application, which means that there is no access to other application features until the dialog box is closed. In order to prevent too many **MsgBox** calls -- in the case of a script not written correctly – VSE keeps track of all function calls demanding user interaction and doesn't show dialog boxes if a customizable limit was exceeded (returns **_MB_OK** in this case).

Example:

```
    …
if( Something )
  {
        …
    str = "Something happened!!!\nShould we continue?"
        result = MsgBox( str ,
         _ MB_YESNOCANCEL | _MB_EXCLAMATION,
                  "Some Title" );

        if( result != _MB_YES )
        ScriptDone();
    … # Go on…
  }
```

19.2 InputBox()

Displays a prompt in a dialog box, waits for the user to input text or click a button, and returns a CSL list object (see the **CSL** manual for details about list objects) or a string containing the contents of the text box.

Format: InputBox(prompt, title, default_text, return_type)

Parameters:

`prompt`	Required. String expression displayed as the message in the dialog box.
`title`	Optional. String expression displayed in the title bar of the dialog box. If you omit **title**, the script name is placed in the title bar.
`default_text`	Optional. String expression displayed in the text box as the default response if no other input is provided. If you omit **default_text**, the text box is displayed empty.
`return_type`	Optional. It specifies the contents of the return object.

The **return_type** argument values are:

Constant	Value	Description
_IB_LIST	0	CSL list object is returned (by Default).
_IB_STRING	1	String input as it was typed in the text box

Return Values:

Depending upon the **return_type** argument, this function returns either a CSL list object or the text typed in the text box as it is.

In case of **return_type = _IB_LIST** (by default), the text in the text box is considered as a set of list items delimited by ',' (only hexadecimal, decimal, and string items are currently supported).

Text example:

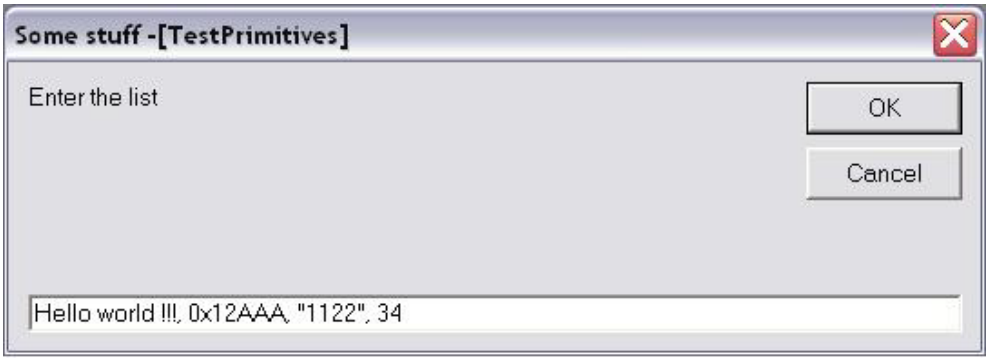
```
Hello world !!!, 12, Something, 0xAA, 10, "1221"
```

Produces a CSL list object of5 items:

```
list =      [ "Hello world !!!", 12, "Something", 0xAA, 10, "1221" ];

list [0] = "Hello world !!!"
list [1] = 12
list [2] = "Something"
list [3] = 0xAA
list [4] = 10
list [5] = "1212"
```

Note: Although the dialog box input text parser tries to determine a type of list item automatically, a text enclosed in quote signs "" is always considered as a string.



Remark:

This function works only for VS Engines controlled via the GUI. For VSEs controlled by COM/Automation clients, it does nothing.

This function "locks" the PCIe Protocol Suite application, which means that there is no access to other application features until the dialog box is closed. In order to prevent too many **InputBox** calls -- in the case of a script not written correctly – VSE keeps track of all function calls demanding user interaction and doesn't show dialog boxes if a customizable limit was exceeded (returns **null** object in that case).

Example:

```
      …
   if( Something )
    {
            …
        v = InputBox( "Enter the list", "Some stuff", "Hello world !!!, 0x12AAA,
Some, 34" );
            ReportText ( FormatEx( "input = %s, 0x%X, %s, %d", v[0],v[1],v[2],v[3] )
);
       … # Go on…

       str =  InputBox( "Enter the string", "Some stuff", "<your string>",
_IB_STRING  );
       ReportText( str );
    }
```

19.3 GetUserDlgLimit()

This function returns the current limit of user dialogs allowed in the verification script. If the script reaches this limit, no user dialogs are shown and the script does not stop. By default, this limit is set to 20.

Format: GetUserDlgLimit()

Example:

```
…
    result = MsgBox( Format( "UserDlgLimit = %d", GetUserDlgLimit() ),
     _MB_OKCANCEL | _MB_EXCLAMATION, "Some Title !!!" );

  SetUserDlgLimit( 2 ); #  set the limit to 2
…
```

19.4 SetUserDlgLimit()

This function sets the current limit of user dialogs allowed in the verification script. If the script reaches this limit, no user dialogs are shown and script does not stop. By default, this limit is set to 20.

Format: SetUserDlgLimit()

Example:

```
…
    result = MsgBox( Format( "UserDlgLimit = %d", GetUserDlgLimit() ),
     _MB_OKCANCEL | _MB_EXCLAMATION, "Some Title !!!" );

  SetUserDlgLimit( 2 ); #  set the limit to 2
…
```

20 String Manipulation/Formating Functions

20.1 FormatEx()

Write formatted data to a string. **FormatEx()** is used to control the way that arguments print out. The format string may contain conversion specifications that affect the way in which the arguments in the value string are returned. Format conversion characters, flag characters, and field width modifiers are used to define the conversion specifications.

Format: **FormatEx (format_string, argument_list)**

Parameters:

`format_string`	Format-control string
`argument_list`	Optional list of arguments to fill in the format string

Return Values:

Formatted string .

Format conversion characters:

Code	Type	Output
c	Integer	Character
d	Integer	Signed decimal integer
i	Integer	Signed decimal integer
o	Integer	Unsigned octal integer
u	Integer	Unsigned decimal integer
x	Integer	Unsigned hexadecimal integer, using "abcdef."
X	Integer	Unsigned hexadecimal integer, using "ABCDEF."
s	String	String

Remark:

A conversion specification begins with a percent sign (**%**) and ends with a conversion character. The following optional items can be included, in order, between the **%** and the conversion character to further control argument formatting:

Flag characters are used to further specify the formatting. There are five flag characters: A minus sign (-) causes an argument to be left-aligned in its field. Without the minus sign, the default position of the argument is right-aligned.

A plus sign (+) inserts a plus sign before a positive signed integer. This only works with the conversion characters **d** and **i**.

A space inserts a space before a positive signed integer. This only works with the conversion characters **d** and **i**. If both a space and a plus sign are used, the space flag is ignored.

A hash mark (#) prepends a 0 to an octal number when used with the conversion character **o**. If # is used with **x** or **X**, it prepends **0x** or 0**X** to a hexadecimal number.

A zero (**0**) pads the field with zeros instead of with spaces.

Field width specification is a positive integer that defines the field width, in spaces, of the converted argument. If the number of characters in the argument is smaller than the field width, then the field is padded with spaces. If the argument has more characters than the field width has spaces, then the field expands to accommodate the argument.

Example:

```
str = "String";
i = 12;
hex_i = 0xAABBCCDD;
…
formatted_str = FormatEx( "%s, %d, 0x%08X", str, i, hex_i );

# formatted_str = "String, 12, 0xAABBCCDD"
```

21 Miscellaneous Functions

21.1 ScriptForDisplayOnly()

Specifies that the script is designed for displaying information only and that its author doesn't care about verification script result. Such a script has a result of **DONE** after execution.

Format: **ScriptForDisplayOnly ()**

Example:

```
ScriptForDisplayOnly();
```

21.2 Sleep()

Asks VSE not to send any events to a script until the timestamp of the next event is greater than the timestamp of the current event plus sleeping time.

Format: Sleep(time)

Parameters:

`time`	VSE time object specifying sleep time

Example:

```
Sleep ( Time(1000) );
# Don't send any event occurred during 1 ms from the current event.
```

21.3 ConvertToHTML()

This function replaces spaces with “ ” and carriage return symbols with “
” in a text string.

Format: ConvertToHTML(text_string)

Parameters:

`text_string`	Text string

Example:

```
str  = "Hello world !!!\n";
str += "How are you today?";

html_str = ConvertToHTML ( str );
# html_string = "Hello&nbspworld !!!<br>How&nbspare&nbspyou&nbsptoday?"
```

Note : Some other useful miscellaneous functions can be found in the file **VSTools.inc.**

21.4 Pause()

Pauses a running script. Later, script execution can be resumed or cancelled.

Format: Pause()

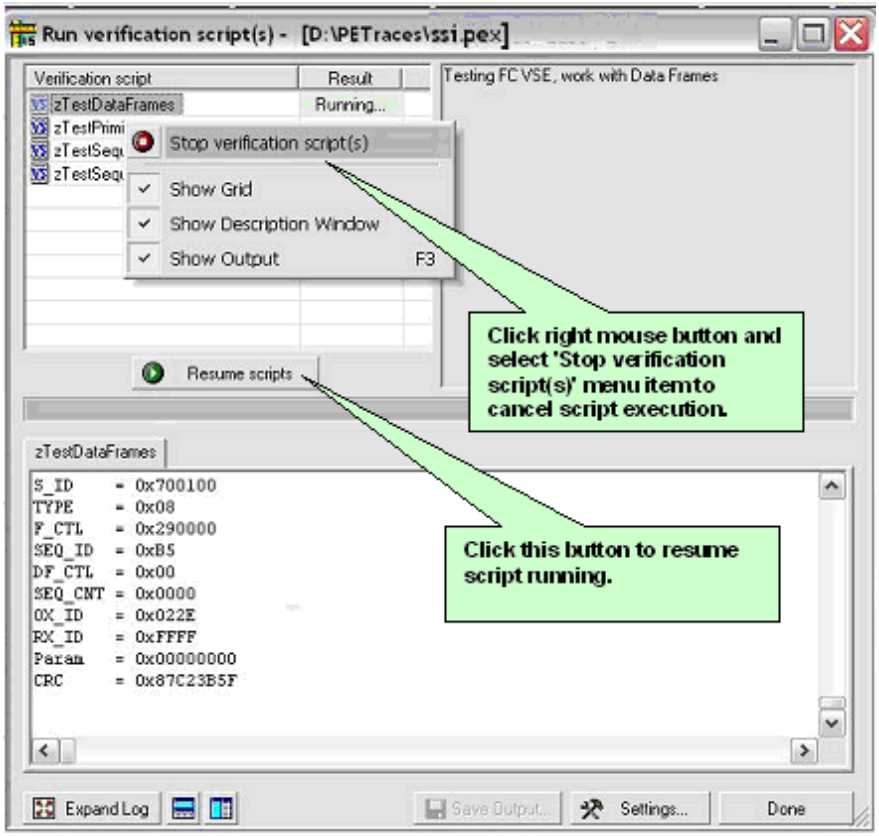
Example:

```
…
If( Something_Interesting() )
{
      GotoEvent(); # Jump to the trace view
      Pause();     # Pause script execution
}
…
```

Remark:

This function works only for VS Engine controlled via a GUI. For VSEs controlled by COM/Automation clients, it does nothing.

When script execution is paused, the Run Verification Script window looks like:



22 The VSE Important Script Files

The VSE working files are located in the **..\Scripts\VFScripts** subfolder of the main PCIe Protocol Suite™ folder. The current version of VSE includes the following files:

File	Description
VSTools.inc	Main VSE file containing definitions of some generic and PCI Express-specific VSE script functions provided by Teledyne LeCroy (must be included in every script). **NOTE**: The files VS_constants.inc and VS_Primitives.inc are included.
VS_constants.inc	File containing definitions of some important generic and PCI Express-specific VSE global constants
VSTemplate.pev_	Template file for new verification scripts.
VSUser_globals.inc	File of user global variable and constant definitions (In this file, it is useful to enter definitions of constants, variables, and functions to be used in many scripts you write.)

22.1 Example Script Files

The VSE example files are located in the **..\Scripts\VFScripts\Samples** subfolder of the main PCIe Protocol Suite folder. The current version of VSE includes the following files:

File	Description
examp_tlp_data.inc	Sample include file containing definitions and functions used by some other sample scripts
examp_dllps.pevs	Sample processing script that outputs information about DLLP packets present in the trace
examp_tlps.pevs	Sample processing script that outputs information about TLP packets present in the trace
examp_ordered_sets.pevs	Sample processing script that outputs information about Ordered Set and Link Condition packets present in the trace
examp_check_errors.pevs	Sample PASS/FAIL script that checks all packets in the trace for all the errors VSE exports and fails in case any error is found
examp_link_transactions.pevs	Sample processing script that outputs information about Link Transactions present in the trace
examp_lm_transaction.pevs	Sample processing script that outputs information about LM transactions present in the trace
examp_ln_transaction.pevs	Sample processing script that outputs information about LN transactions present in the trace
examp_split_transactions.pevs	Sample processing script that outputs information about Split Transactions present in the trace
examp_mctp_cmd.pevs	Sample processing script that outputs information about MCTP Command transactions present in the trace
examp_mctp_msg.pevs	Sample processing script that outputs information about MCTP Message transactions present in the trace
examp_metrics.pevs	Sample processing script that outputs information about Memory Write Link Transaction metrics and all Split Transaction metrics
examp_nvme.pevs	Sample processing script that outputs information about NVM Transactions present in the trace
examp_nvme_errors.pevs	Sample processing script that outputs information about NVM Transaction errors present in the trace
examp_nvme_cmd.pevs	Sample processing script that outputs information about NVM Commands present in the trace
examp_nvme_cmd_errors.pevs	Sample processing script that outputs information about NVM Commands errors present in the trace

File	Description
examp_nvme_cmd_deltatime_metrics.pevs	Sample processing script that outputs information about NVM Commands delta time metrics present in the trace
examp_ahci.pevs	Sample processing script that outputs information about AHCI Transactions present in the trace
examp_ata.pevs	Sample processing script that outputs information about ATA Transactions present in the trace
examp_ahci_errors.pevs	Sample processing script that outputs information about AHCI Transaction errors present in the trace
examp_ata_errors.pevs	Sample processing script that outputs information about ATA Transaction errors present in the trace
examp_mctp.pevs	Sample processing script that outputs information about MCTP packets sent with Vendor Defined Messages
examp_mctp_messages.pevs	Sample processing script that outputs information about MCTP Messages
examp_pldm.pevs	Sample processing script that outputs information about some of PLDM Messages
examp_smbus.pevs	Sample processing script that outputs information about SMBus packets present in the trace.
examp_nvmc_admin_commands.pevs	Sample processing script that outputs information about NVMe Admin commands.
examp_nvmc_get_log_page.pevs	Sample processing script that outputs information about NVMe Get Log Page commands.
examp_nvmc_identify.pevs	Sample processing script that outputs information about NVMe Identify commands.
examp_nvmc_set_features.pevs	Sample processing script that outputs information about NVMe Set Features commands.
examp_nvme_submission_dptr.pevs	Sample processing script that outputs information about NVMe submission queue entry dptr field.
examp_cxl_almp_nullflit.pevs	Sample processing script that outputs information about CXL ALMP and NULL Flit packets present in the trace
examp_cxl_cache_mem.pevs	Sample processing script that outputs information about CXL Cache and CXL Mem packets present in the trace
examp_cxl_io_tlps.pevs	Sample processing script that outputs information about CXL VDM TLPs packets present in the trace

Appendix A How to Contact Teledyne LeCroy

Send e-mail…	psgsupport@teledynelecroy.com
Contact support…	teledynelecroy.com/support/contact
Visit Teledyne LeCroy's web site…	teledynelecroy.com
Tell Teledyne LeCroy…	Report a problem to Teledyne LeCroy Support via e-mail by selecting **Help > Tell Teledyne LeCroy** from the application toolbar. This requires that an e-mail client be installed and configured on the host machine.