

Project Zipline Top

Micro Architecture Specification

**Authors:**

**Microsoft Corporation**

**Broadcom Corporation**

**Revision History**

|  |  |
| --- | --- |
| Date | Description |
| 04/03/2019 | Version 1.0 |
|  |  |

**License**

Contributions to this Specification are made under the terms and conditions set forth in the Open Web Foundation Contributor License Agreement (“OWF CLA 1.0”) (“Contribution License”) by:

Microsoft Corporation

Broadcom Corporation

Usage of this Specification is governed by the terms and conditions set forth in Open Web Foundation Final Specification Agreement (“OWFa 1.0”) (“Specification License”).

**Note**: The following clarifications, which distinguish technology licensed in the Contribution License and/or Specification License from those technologies merely referenced (but not licensed), were accepted by the Incubation Committee of the OCP:

**None.**

NOTWITHSTANDING THE FOREGOING LICENSES, THIS SPECIFICATION IS PROVIDED BY OCP "AS IS" AND OCP EXPRESSLY DISCLAIMS ANY WARRANTIES (EXPRESS, IMPLIED, OR OTHERWISE), INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR A PARTICULAR PURPOSE, OR TITLE, RELATED TO THE SPECIFICATION. NOTICE IS HEREBY GIVEN, THAT OTHER RIGHTS NOT GRANTED AS SET FORTH ABOVE, INCLUDING WITHOUT LIMITATION, RIGHTS OF THIRD PARTIES WHO DID NOT EXECUTE THE ABOVE LICENSES, MAY BE IMPLICATED BY THE IMPLEMENTATION OF OR COMPLIANCE WITH THIS SPECIFICATION. OCP IS NOT RESPONSIBLE FOR IDENTIFYING RIGHTS FOR WHICH A LICENSE MAY BE REQUIRED IN ORDER TO IMPLEMENT THIS SPECIFICATION. THE ENTIRE RISK AS TO IMPLEMENTING OR OTHERWISE USING THE SPECIFICATION IS ASSUMED BY YOU. IN NO EVENT WILL OCP BE LIABLE TO YOU FOR ANY MONETARY DAMAGES WITH RESPECT TO ANY CLAIMS RELATED TO, OR ARISING OUT OF YOUR USE OF THIS SPECIFICATION, INCLUDING BUT NOT LIMITED TO ANY LIABILITY FOR LOST PROFITS OR ANY CONSEQUENTIAL, INCIDENTAL, INDIRECT, SPECIAL OR PUNITIVE DAMAGES OF ANY CHARACTER FROM ANY CAUSES OF ACTION OF ANY KIND WITH RESPECT TO THIS SPECIFICATION, WHETHER BASED ON BREACH OF CONTRACT, TORT (INCLUDING NEGLIGENCE), OR OTHERWISE, AND EVEN IF OCP HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Contents

[1 Terminology 7](#_Toc5185105)

[2 Overview 9](#_Toc5185106)

[2.1 CCEIP Block Diagram and DataFlow 11](#_Toc5185107)

[2.1.1 CCEIP DataFlow 11](#_Toc5185108)

[2.1.2 Inbound Streaming FIFO 11](#_Toc5185109)

[2.1.3 CRC Generator and Checker 12](#_Toc5185110)

[2.1.4 Prefix Engine 12](#_Toc5185111)

[2.1.5 Prefix Attach 12](#_Toc5185112)

[2.1.6 LZ77 Compressor 12](#_Toc5185113)

[2.1.7 Huffman Encoder 13](#_Toc5185114)

[2.1.8 Crypto Encryption Engine 13](#_Toc5185115)

[2.1.9 CRC-C/CRC-G 14](#_Toc5185116)

[2.1.10 Outbound Streaming Interface Formatter (OSF) 14](#_Toc5185117)

[2.1.11 Decryption Engine 14](#_Toc5185118)

[2.1.12 Decompressor 14](#_Toc5185119)

[2.1.13 Authentication Engine (Crypto Block) 15](#_Toc5185120)

[2.1.14 CRC Checker 15](#_Toc5185121)

[2.1.15 Completion Formatter 15](#_Toc5185122)

[2.2 CDDIP Block Diagram and Dataflow 15](#_Toc5185123)

[2.2.1 CDDIP DataFlow 16](#_Toc5185124)

[2.2.2 Inbound Streaming FIFO 16](#_Toc5185125)

[2.2.3 CRC Checker 17](#_Toc5185126)

[2.2.4 Decryption Engine 17](#_Toc5185127)

[2.2.5 Prefix Attach Engine 17](#_Toc5185128)

[2.2.6 Decompressor 17](#_Toc5185129)

[2.2.7 Authentication Engine (Crypto Block) 17](#_Toc5185130)

[2.2.8 CRC Generator 18](#_Toc5185131)

[2.2.9 Completion Formatter 18](#_Toc5185132)

[2.2.10 Outbound Streaming Interface Formatter (OSF) 18](#_Toc5185133)

[3 CCEIP and CDDIP Data Movement and Control Processing. 19](#_Toc5185134)

[3.1 Project Zipline TLV Headers 19](#_Toc5185135)

[3.1.1 RQE TLV Data Message 20](#_Toc5185136)

[3.1.2 AUX Command TLV 23](#_Toc5185137)

[3.1.3 AUX FRMD IN TLV 33](#_Toc5185138)

[3.1.4 Data TLV 41](#_Toc5185139)

[3.1.5 CQE TLV 43](#_Toc5185140)

[3.1.6 SCH TLV 50](#_Toc5185141)

[3.2 Project Zipline Command Types and Allowable Header Stack ups 53](#_Toc5185142)

[3.2.1 Top Level Interfaces 55](#_Toc5185143)

[4 Detailed Block Flow Walkthrough 59](#_Toc5185144)

[4.1.1 CCEIP Command Processing 59](#_Toc5185145)

[4.1.2 CDDIP Command Processing 84](#_Toc5185146)

[4.1.3 Error Reporting/Handling 92](#_Toc5185147)

[4.1.4 Debug Features 92](#_Toc5185148)

[4.1.5 AUX\_CMD Debug Field Usage 92](#_Toc5185149)

[5 Datapath Bandwidth Analysis 94](#_Toc5185150)

Table of Figures

[Figure 1 : CCEIP Top Level Block Diagram 14](#_Toc3986294)

[Figure 2 : CDDIP Top Level Block Diagram 19](#_Toc3986295)

[Figure 3 : Allowable Header Stack ups for Simple Command 56](#_Toc3986296)

[Figure 4: Allowable Header Stackups for Compound Commands 57](#_Toc3986297)

[Figure 5 : ISF Simple Command TLV Manipulation 63](#_Toc3986298)

[Figure 6: RawData Checksum Generator/Checker TLV Manipulation 65](#_Toc3986299)

[Figure 7: Prefix Engine TLV Manipulation (Simple) 66](#_Toc3986300)

[Figure 8: Prefix Attach TLV Manipulation 68](#_Toc3986301)

[Figure 9: LZ77 Compressor TLV Manipulation 69](#_Toc3986302)

[Figure 10: Huffman Encoder TLV Manipulation 70](#_Toc3986303)

[Figure 11: Encryption and Authentication TLV Manipulation (Simple Commands) 72](#_Toc3986304)

[Figure 12: Encrypted/Compressed Checksum Generator 73](#_Toc3986305)

[Figure 13: Encrypted/Compressed Checksum Checker 74](#_Toc3986306)

[Figure 14: Decryption and Authentication Validation Engine TLV Manipulation 75](#_Toc3986307)

[Figure 15: Huffman Decoder TLV Manipulation 75](#_Toc3986308)

[Figure 16: LZ77 TLV Manipulation 77](#_Toc3986309)

[Figure 17: Raw CRC Checker TLV Manipulation 78](#_Toc3986310)

[Figure 18: Raw Integrity and Authentication Engine 79](#_Toc3986311)

[Figure 19: Completion Streaming Interface 80](#_Toc3986312)

[Figure 20 : ISF Simple Command TLV Manipulation 89](#_Toc3986313)

[Figure 21: Encrypted/Compressed Checksum Checker 89](#_Toc3986314)

[Figure 22: Decryption and Authentication Validation Engine TLV Manipulation 90](#_Toc3986315)

[Figure 23: Prefix Attach TLV Manipulation 91](#_Toc3986316)

[Figure 24: Huffman Decoder TLV Manipulation 92](#_Toc3986317)

[Figure 25: LZ77 TLV Manipulation 93](#_Toc3986318)

[Figure 26: Raw Integrity and Authentication Engine 94](#_Toc3986319)

[Figure 27: Raw CRC Generator TLV Manipulation 94](#_Toc3986320)

Table of Tables

[Table 1: Cryptography Algorithms Supported 13](#_Toc3986610)

[Table 2: Authentication Algorithms Supported 14](#_Toc3986611)

[Table 3: Prefix Engine TLV Manipulation (Compound Command) 64](#_Toc3986612)

[Table 4: Prefix Attach Engine Command Processing 64](#_Toc3986613)

[Table 5: Footer Fields at Completion Streaming Interface 79](#_Toc3986614)

[Table 6: Error Codes 83](#_Toc3986615)

# Terminology

|  |  |
| --- | --- |
| TLV | Type-Length-Value Headers  The CCEIP/CDDIP carry frame and command information that are bound together by a standard “TLV” header that describes the information (type), the length of the data and the actual data contents (value). This format allows engines to skip over information not intended for them. |
| MIC | Message Integrity Check  Fields of this type protect data from random bit errors or re-assembly errors. |
| MAC | Message Authentication Check  Fields of this type protect data from man-in the middle attacks or other spoofing. They rely on secret keys to make transformations on the data. They also can provide protections against random bit errors or re-assembly errors. |
| SHA | Secure Hash Algorithm  A crypto graphic hash function that creates a 32Byte signature for text. |
| AES | Advanced Encryption Standard  A symmetric encryption algorithm that utilizes 128 bit blocks of data and keys of 128/192/256 bits. |
| LZ77 | Lempel-Ziv ’77 Compression is a single pass compression scheme which replaces repeating text data with a backward referencing pointer and length. |
| HistoryBuffer | The amount of data buffered by the LZ77 engine from which to draw backward references and lengths. |
| Prefix | Data that is prepended to the start of a frame of compressible data that doesn’t actually get encrypted, but is used to “seed” the history buffer so as to allow the beginning of the frame to draw upon data to build pointers. |
| CRC | Cyclic Redundancy Checksum  An error detected code. |
| ISM | Intra-Stage Monitor  A common building block used in the CCEIP/CDDIP which sits between major interfaces used for debug capability. This block has a small history buffer (fifo) along with triggering and backpressure capabilities to use for verification and in-circuit debug. |
| TLV Parser | A common building block in the CCEIP/CDDIP which interrogates the common datapath and delineates the TLV headers. This block has debug capability to corrupt the frame data, but is mostly used to allow an engine sub-module to be built and TLV headers which it doesn’t care about to be “skipped”. |
| BIMC | Built-In-Memory Controller  This is a BRCM standard ECC controller which is used to provide a harness around internal SRAMs. It can monitor ECC errors as well as inject errors for test. It functions as a serial daisy chain that connects memories in a block to a standard top-level controller. |
| Key | A field used as input to the crypto engine to encrypt or decrypt data. |
| Deflate | A common compression scheme based on LZ77 and a standardized Huffman Symbol Mapping/Dictionary. |
| XP9 | A MSFT proprietary LZ77+Huffman Compression Scheme based on a 64KB History Buffer and a pair of alphabets (Short + Long) used in the Huffman Encoder. |
| XP10 | A MSFT Proprietary LZ77+ Huffman Compression Scheme similar to XP9 but adds Prefixs as well as a smaller overhead for the headers. |
| XP10CFH | A MSFT Proprietary LZ77 + Huffman Compresison Scheme, utilizing the features of XP10, but with reduced overhead. |

# Overview

The Project Zipline CCEIP and CDDIP engines provide compression and cryptography services. This is accomplished by a pair of engines:

* **C**ompression **E**ncryption **IP** (**CCEIP**)

This engine compresses and encrypts frames of data as well as provides validation services (by decompressing and decrypting the frame and comparing against CRCs).

* **D**ecompression **D**ecryption **IP** (**CDDIP**)

This engine decrypts and decompresses frames of data.

Project Zipline has 4 CCEIP engines and 4 CDDIP engines. Each individual engine is capable of processing 25Gbps of traffic when using 2KB frames. These engines have the following requirements:

* Compression (LZ77 Based)
  + Deflate Compatible (limit matches to 258Bytes)
  + XP9/XP10 (matches limited only by size of the history buffer)
    - XP9 Level 6 (64KB History Buffer supported)
    - XP10 Level 6 (64 KB History Buffer supported)
      * XP9/XP10 Supports 4 Move to Front
  + Dynamic (Retrospective) Huffman Compression Support
    - Deflate
    - XP9/XP10
  + Predefined Huffman Table Support for XP10
  + Predefined Prefix and User Prefix Support for XP10
  + Raw Mode option when compressor inflates size of packet.
* Cryptography Algorithms Supported:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Mode | Key Size | IV/Tweak | AAD | Standard | Notes |
| NULL | NA | NA | NA | -- | No encryption or authentication (bypass) |
| AES-XEX-512 | 2x256 (key1, key2) | 128b | NA | IEEE 1619-2007  [NIST 800-38e](http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38e.pdf) | Must be padded to cipher block |
| AES-XTS-512 | 2x256 (key1, key2) | 128b | NA | IEEE 1619-2007  [NIST 800-38e](http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38e.pdf) |  |
| AES-GCM-256 | 256 key | 96b | 0-255B | [NIST SP800-38d](http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf) | 128 Authentication tag |
| AES-GMAC-256 | 256 key | 96b | 0-255B | [NIST SP800-38d](http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf) | No encryption, auth only. 128b Authentication Tag |

* Authentication Algorithms Supported:

|  |  |  |  |
| --- | --- | --- | --- |
| Mode | Specification | Key Size | Digest Size |
| NULL | NA | NA | 0 |
| SHA-256 | FIPS 180-4 | NA | 256 |
| SHA-256C | FIPS 180-4 | NA | 64\* |
| HMAC-SHA-256 | FIPS 198-1 | 256 | 256 |
| HMAC-SHA-256C | FIPS 198-1 | 256 | 64\* |

It is expected that readers of this spec are fluent in:

* AXI Streaming Interface and AMBA nomenclature
* LZ77 Compression/Decompresion
* Huffman Encoding/Decoding
* Deflate/GZIP
* Cryptography and Authentication

## CCEIP Block Diagram and DataFlow

Below is a top level block diagram for the CCEIP; as can be seen the CCEIP contains the compression and encryption engines, but also contains decryptor and decompressor to be used as a validator to ensure there were no errors during processing of the compressed/encrypted data – otherwise what will be written to disk will not be retrievable.

This section captures a very high level command processing flow to get a basic understanding of the blocks, a more detailed flow walkthrough is presented later in this chapter for the Simple and Compound Command processing.



Figure : CCEIP Top Level Block Diagram

### CCEIP DataFlow

A Command and its associated frame data will be fed into the CCEIP over an AXI Streaming interface driven by the TXC.

* All **Data and Metadata** in and out of the engines is specified as Little Endian
  + Specifically, on the 64 bit AXI Interface the first byte of the frame will be located on bits [7:0] of the databus. The Verilog AXI Interface, to transmit the incrementing pattern of 0 to 7 would be :
    - Axi\_wdata[63:0] = 64’h07\_06\_05\_04\_03\_02\_01\_00

### Inbound Streaming FIFO

* Incoming data is fed to a front-end decoupling fifo sized at 12KB. The Fifo controls dataflow using AXI Streaming Valid/Ready handshaking. The fifo will have a programmable watermark to account for top level pipelining delays on the datapath. A series of headers called “TLVs” will be presented with the frame data (in fact the data itself is a TLV); these TLVs will be manipulated and examined by the downstream blocks.
* This block does the following:
  + Does any manipulation of the front end headers that the TXC is / was unable to do. This involves:
    - Converting the incoming Data TLV into a User Prefix TLV and Data TLV by segmenting based on the CMD.Compression header Prefix fields.
  + Provide several programmable “triggers” which will allow the user to effectively “Single-Step” the design by playing data out of the Inbound Fifo via a debug interface. These triggers would allow the user to specify a breakpoint based on the header and frame data that enters the block.
  + Provide for basic Frame/Byte Counters on the inbound traffic.
  + Modulate the AXI Streaming “ready” interface signal back to the Project Zipline TXC so as to test backpressure.

### CRC Generator and Checker

This engine builds a CRC over the incoming payload data using a selectable polynomial (CRC-32 or CRC64) and places this information in an TLV.

* Generates a CRC32 or CRC64 over the Frame.

### Prefix Engine

The Prefix Engine examines the first 4KB of the packet and will determine if the frame is recognized as being of a particular class or type. The Prefix engine places this information in a special field in a TLV for use by the downstream blocks.

### Prefix Attach

The Prefix Attach Engine examines the TLVs and if it detects that the Prefix Engine has found a valid prefix, it will add up to 2 additional TLVs to the Frame:

1. Prefix Data (1KB) which is provided from 64 possible on-chip prefixes.
2. Predetermined Huffman Tree which is provided from 64 possible on-chip trees

### LZ77 Compressor

The LZ77 compressor has a 64K sliding window and logic to generate LZ77 Symbols based on 4 byte n-grams. The throughput of this engine is to be 4 characters per clock (4 Bytes \* 800MHz ~ 25Gbps).

* + In addition to basic LZ77 Symbol Generation, the compressor also supports up to 4 special “Move To Front” entries which allow LZ77 Symbols to reference the last 4 transmitted symbols. This has the effect of “spiking” the Huffman Encoder and increasing the frequency of particular symbols, thus allowing smaller symbol lengths to be used.
  + With the exception of XP9, the operation of the LZ77 Compressor is the same for all of the other supported compression algorithms. The difference for XP9 has to do with the way the MTF list is handled as mandated by the XP9 specification.

### Huffman Encoder

The Huffman Encoder has the capability of running in the following modes and generating data for the following formats:

* + XP9 Encoding
    - Retrospective Huffman Encoding, maximum # of Symbols 8K
    - Raw Data (due to Huffman Encoding being larger)
  + XP10 Encoding/CFH Modes
    - Retrospective Huffman Encoding, maximum # of Symbols 8K
    - Raw Data (due to Huffman Encoding being larger)
    - Predetermined Huffman Encoding, maximum #of Symbols 8K
  + GZIP (Deflate)
    - Retrospective Huffman Encoding, using Deflate mapping
    - Raw Data (due to Huffman Encoding being larger)
  + ZLIB (Deflate)
    - Retrospective Huffman Encoding, using Deflate mapping
    - Raw Data (due to Huffman Encoding being larger)

#### Schedule Update Interface

The Schedule Update interface will supply inflation/deflation information back to the centralized scheduler; this interface will be updated by the compressor on Huffman Coding block boundaries and on decompression at Coding block boundaries. The information passed on this interface is:

* Size of frame, measured in rounded up bytes subject to 2 additional adjustments.
  + Adjustment #1 : Global Byte Adjustment
  + Adjustment #2 : Prefix Adjustment.

### Crypto Encryption Engine

A Cryptography and Authentication Engine is used to provide 25Gbps services to the CCEIP and CDDIP engines This block has the following modes of operation:

Table 1: Cryptography Algorithms Supported

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Mode | Key Size | IV/Tweak | AAD | Standard | Notes |
| NULL | NA | NA | NA | -- | No encryption or authentication (bypass) |
| AES-XEX-512 | 2x256 (key1, key2) | 128b | NA | IEEE 1619-2007  [NIST 800-38e](http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38e.pdf) | Must be padded to cipher block |
| AES-XTS-512 | 2x256 (key1, key2) | 128b | NA | IEEE 1619-2007  [NIST 800-38e](http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38e.pdf) | Cipher text stealing used |
| AES-GCM-256 | 256 key | 96b | 0-255B | [NIST SP800-38d](http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf) | 128 Authentication tag |
| AES-GMAC-256 | 256 key | 96b | 0-255B | [NIST SP800-38d](http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf) | No encryption, auth only. 128b Authentication Tag |

Table 2: Authentication Algorithms Supported

|  |  |  |  |
| --- | --- | --- | --- |
| Mode | Specification | Key Size | Digest Size |
| NULL | NA | NA | 0 |
| SHA-256 | [FIPS.180-4](https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf) | NA | 256 |
| AES-GMAC | [NIST SP800-38d](http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf) | 256 | 128 |
| HMAC-SHA-256 | [FIPS.198-1](https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.198-1.pdf) | 256 | 256 |

The Crypto Engine receives KEY material on a dedicated sideband interface. This engine will encrypt data as well as optionally add a SHA2/HMAC-SHA-256 digest to a TLV.

### CRC-C/CRC-G

This CRC Generator can optionally build a CRC32/CRC64 over the compressed and encrypted data and place this CRC in a TLV.

### Outbound Streaming Interface Formatter (OSF)

At this point in the CCEIP pipeline, the datapath bifurcates. The compressed / encrypted data is sent to the outbound interface formatter, which readies the data for transmission to the RXC engine.

The Outbound Interface formatter contains an 8KB FIFO to decouple the bursty nature of the traffic.

### Decryption Engine

This is a second instance of the Crpyto Engine, wired into Decrypt mode. It will decrypt and authenticate the data.

### Decompressor

The decompressor engine is designed to accept a datastream in excess of 25Gbps. Since the packet data will be inflated on its output side, performance is limited to the outgoing 64 bit datapath (50Gbps) as well as any system based backpressure. The Decompressor is designed to run without stalls on a 2KB Packet. This implies that the overhead of building a Huffman table is less than 512 clock cycles (25Gbps/800MHz = 512 cycles).

### Authentication Engine (Crypto Block)

An authentication engine will be placed after the decompressor which is a special build option of the Crypto block previously used which does not contain the AES engine (so it is smaller). This block will build a SHA digest over the Raw data and place this in a TLV.

### CRC Checker

The decompressed, decrypted data will be subject to a final check where the original incoming CRC will be checked

### Completion Formatter

The Completion formatter generates a message to send to the enqueuer/dequeuer engines and inform them of the status of the requested compression/encryption operation.

## CDDIP Block Diagram and Dataflow

Below is a top level block diagram for the CDDIP; as can be seen the CDDIP contains the decompression and decryption engines.

This section captures a very high level command processing flow to get a basic understanding of the blocks, a more detailed flow walkthrough is presented later in this chapter for the Simple and Compound Command processing.



Figure : CDDIP Top Level Block Diagram

### CDDIP DataFlow

A Command and its associated frame data will be fed into the CCEIP over an AXI Streaming interface driven by the TXC.

* All **Data and Metadata** in and out of the engines is specified as Little Endian
  + Specifically, on the 64 bit AXI Interface the first byte of the frame will be located on bits [7:0] of the databus. The Verilog AXI Interface, to transmit the incrementing pattern of 0 to 7 would be :
    - Axi\_wdata[63:0] = 64’h07\_06\_05\_04\_03\_02\_01\_00

### Inbound Streaming FIFO

* Incoming data is fed to a front-end decoupling fifo sized at 12KB. The Fifo controls dataflow using AXI Streaming Valid/Ready handshaking. The fifo will have a programmable watermark to account for top level pipelining delays on the datapath. A series of headers called “TLVs” will be presented with the frame data (in fact the data itself is a TLV); these TLVs will be manipulated and examined by the downstream blocks.
* This block does the following:
  + Does any manipulation of the front end headers that the TXC is / was unable to do. This involves:
    - Converting the incoming Data TLV into a User Prefix TLV and Data TLV by segmenting based on the CMD.Compression header Prefix fields.
  + Provide several programmable “triggers” which will allow the user to effectively “Single-Step” the design by playing data out of the Inbound Fifo via a debug interface. These triggers would allow the user to specify a breakpoint based on the header and frame data that enters the block.
  + Provide for basic Frame/Byte Counters on the inbound traffic.
  + Modulate the AXI Streaming “ready” interface signal back to the Project Zipline TXC so as to test backpressure.

### CRC Checker

Calculates a CRC across the frame data and compares it to a CRC in the FOOTER TLV.

### Decryption Engine

This is an instance of the Crpyto Engine as described in section 2.1.8, wired into Decrypt mode. It will decrypt and authenticate the data. In this case, authentication means that this engine builds a SHA digest over the encrypted data and compares it against the Cipher Data MAC in the Footer.

### Prefix Attach Engine

The Prefix Attach Engine examines the TLVs and if it detects that the Prefix Engine has found a valid prefix, it will add up to 2 additional TLVs to the Frame:

1. Prefix Data (1KB) out of 64 possible on-chip prefixes.
2. Predetermined Huffman Tree

### Decompressor

The decompressor engine is designed to accept a datastream in excess of 25Gbps. Since the packet data will be inflated on its output side, performance is limited to the outgoing 64 bit datapath (50Gbps) as well as any system based backpressure. The Decompressor is designed to run without stalls on a 2KB Packet. This implies the otherhead of building a Huffman table is less than 512 clock cycles (25Gbps/800MHz = 512 cycles).

### Authentication Engine (Crypto Block)

An authentication engine will be placed after the decompressor which is a special build option of the Crypto block previously used which does not contain the AES engine (so it is smaller). This block will build a SHA digest over the Raw data and place this in a TLV.

### CRC Generator

This CRC Generator can optionally build a CRC32/CRC64 over the decompressed and decrypted data and place this CRC in a TLV.

### Completion Formatter

The Completion formatter generates a message to send to the enqueuer/dequeuer engines and inform them of the status of the requested compression/encryption operation. If an error was detected in the decompressed/decrypted datastream the compressed information will be discarded and software may choose to restart the operation.

### Outbound Streaming Interface Formatter (OSF)

At this point in the CCEIP pipeline, the datapath bifurcates. The compressed / encrypted data is sent to the outbound interface formatter, which readies the data for transmission to the RXC engine.

The Outbound Interface formatter contains an 8KB FIFO to decouple the bursty nature of the traffic.

# CCEIP and CDDIP Data Movement and Control Processing.

The three Project Zipline engines have a unified datapath/control path that allows for in-band control information to be passed on a highspeed transport bus that is shared with the datapath. DataPath can be the main high speed datapath or the separate Key Delivery bus used to carry sensitive cryptographic data.

In order for the engines to understand what is control information and what is data information, a series of headers have been constructed to allow the blocks to interoperate and segregate processing control. These headers are called the Project Zipline TLV Headers. The term TLV is used to describe Type, Length and Value.

## Project Zipline TLV Headers

The Project Zipline Pipeline engines will receive a variable length header stack up that is encoded as “TYPE/Length/Value” style headers. These headers are standardized so that they all have the same base composition and can be easily parsed by sub-blocks. The TLVs have the following properties:

1. TLVs are always 8 Byte aligned, which allows for simpler parsing in the Project Zipline 64 bit datapaths.
2. The first word (64 bits of every TLV) has the same structure and the following elements in the same locations:
   1. Type
   2. Length
   3. CRC Protection of this beat

The list of Project Zipline TLV headers that have currently been defined is as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **TLV Type #** | **Length (in bytes)** | **CCEIP/CDDIP**  **Internal**  **TLV**  **Only** |
| **RQE TLV**  The RQE TLV is the first header that the PROJECT ZIPLINE IP will receive for command; it contains information that is built directly into the RQE. | 0 | 16 |  |
| **CMD TLV**  The CMD TLV is the second header in the stackup and controls the general PROJECT ZIPLINE IP block processing. It is received once per | 1 | 24 |  |
| **KEY TLV**  This TLV is used to convey KEY information and KEY Operations processing information to the CCEIP and CDDIP. | 2 | 160 |  |
| **PHD TLV**  This TLV carries the Predetermined Huffman tables | 3 | 536 | Yes |
| **PFD TLV**  This TLV carries the Prefix Data (User and Predetermined) | 4 | 1KB-64KB | Yes |
| **DATA (Known Size) TLV** | 5 | 4B-8MB |  |
| **DATA (Unknown Size) TLV** | 19 | 4B-8MB |  |
| **GUID** | 10 | 40 |  |
| **FOOTER** | 6 | 112/160 | Yes |
| **CQE TLV** | 9 | 16 |  |
| **STATS TLV** | 8 | 24 |  |
| **LZ77-SYM TLV** | 7 | Variable |  |
| **FRMD Types** | 11-18  22 | Various |  |
| **SCH TLV** | 26 | 20 |  |

**Note**

Verilog structs for each 64-bit word of the TLVs are provided in the following file of the code release:

Project Zipline/rtl/common/include/cr\_structs.sv

### RQE TLV Data Message

This TLV indicates the start of an operation. It carries the information that is passed (or inferred) from the RQE.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Offset*** | ***31*** |  | | | | | | ***0*** |
| 0x0 | unused(4) | | | [eng](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#rqe_tlv/NONE/NONE/eng) | [seq](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#rqe_tlv/NONE/NONE/seq) | | [length](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#rqe_tlv/NONE/NONE/length) | [type](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#rqe_tlv/NONE/NONE/type) |
| 0x4 | [bip2](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#rqe_tlv/NONE/NONE/bip2) | [no\_data](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#rqe_tlv/NONE/NONE/no_data) | [aux\_frmd\_crc](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#rqe_tlv/NONE/NONE/aux_frmd_crc) | [frame\_size](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#rqe_tlv/NONE/NONE/frame_size) | [vf\_valid](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#rqe_tlv/NONE/NONE/vf_valid) | [trace](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#rqe_tlv/NONE/NONE/trace) | unused(11) | [frame](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#rqe_tlv/NONE/NONE/frame) |
| 0x8 | [src\_data\_len](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#rqe_tlv/NONE/NONE/src_data_len) | | | | | | | |
| 0xc | [pf](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#rqe_tlv/NONE/NONE/pf) | | | [vf](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#rqe_tlv/NONE/NONE/vf) | | | [scheduler\_handle](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#rqe_tlv/NONE/NONE/scheduler_handle) | |

**Bits: 113 / Bytes: 15 / Unused Bits: 15**

#### type (Offset:0x0[7:0], Size: 8)

TLV Type - The value in this field defines the field format to apply to the entire TLV.

|  |  |  |
| --- | --- | --- |
| ***Value*** | ***Enum*** | ***Enumeration Description*** |
| 0 | rqe | RQE TLV |

#### length (Offset:0x0[15:8], Size: 8)

Length of the TLV in units of 4 bytes. This length is computed over the entirety of the TLV Type, Length and data fields.

The length may indicate that only part of the data in the last beat is valid. The next TLV will start at bit zero of the interface. Unused area in the last beat will be sent as zero.

#### seq (Offset:0x0[23:16], Size: 8)

Per-Engine sequence number for the operation. This value will be incremented by one for each operation on an engine. All TLV for the same operation will have the same sequence number.

#### eng (Offset:0x0[27:24], Size: 4)

Engine ID for the operation.

#### frame (Offset:0x4[10:0], Size: 11)

Frame number within operation. This field will be zero for TLV that occur just once within the command.

#### trace (Offset:0x4[22], Size: 1)

When this bit is '1', it indicates that this operation had the trace bit set in the RQE. Processing blocks should fill their trace registers during the processing of this command.

#### vf\_valid (Offset:0x4[23], Size: 1)

When this bit is '1', the vf field is valid and the operation is for a VF. When this bit is zero, the operation is for a PF.

#### frame\_size (Offset:0x4[27:24], Size: 4)

Frame size for command. This defines the frame size for the command.

|  |  |  |
| --- | --- | --- |
| ***Value*** | ***Enum*** | ***Enumeration Description*** |
| 0 | SIMPLE | Command will be executed as a single frame up to 8MB. |
| 5 | COMPOUND\_4K | Command will be broken up into 4KB frames. |
| 6 | COMPOUND\_8K | Command will be broken up into 8KB frames. |

#### aux\_frmd\_crc (Offset:0x4[28], Size: 1)

When this bit is '1', the RXC will generate a CRC32 over the AUX\_FRAME\_DATA area that is written back to the host. The engines will pass this value through so RXC can do the operation.

#### no\_data (Offset:0x4[29], Size: 1)

Frame can be zero length operations without errors or with errors.

When this bit is asserted into the engine, the data TLV will be for one byte and the other TLVs will be as defined for an "no\_data" operation.

The engine will operate on the no\_data case as normal.

This bit is passed through to RXC from TXC. The purpose of this bit is to tell RXC to ignore and drop the data TLV and aux\_frmd TLV for the command as the inputs were manufactured by TXC in the absense of valid data from the host.

#### bip2 (Offset:0x4[31:30], Size: 2)

2bit parity value over the first 64b of the TLV. bip2[1] contains xor computed over the odd bits in the first 64b word of TLV excluding bip2[1]. bip2[0] contains xor computed over the even bits in the first 64b word of TLV excluding bip2[0].

#### src\_data\_len (Offset:0x8[31:0], Size: 32)

Length of data buffer for command including any optional User Prefix Data.

#### scheduler\_handle (Offset:0xc[15:0], Size: 16)

|  |  |  |
| --- | --- | --- |
| ***Offset*** | ***Field*** | ***Field Description*** |
| 15:13 | qg\_cre | This field carries the queue group compression ratio estimate.  This field is used to scale the initial outbound bytes charge in TWE, and the final outbound bytes refund in RXC.  For compression, the scale factor represents a negative exponent. I.e., the scale factor will be 2^(-qg\_cre).  For decompression, the scale factor represents a positive exponent. I.e., the scale factor will be 2^qg\_cre.   |  |  |  | | --- | --- | --- | | ***Value*** | ***Enum*** | ***Enumeration Description*** | | 0 | sf1 | Compression scale factor of 1.  Decompression scale factor of 1. | | 1 | sf2 | Compression scale factor of 1/2.  Decompression scale factor of 2. | | 2 | sf4 | Compression scale factor of 1/4.  Decompression scale factor of 4. | | 3 | sf8 | Compression scale factor of 1/8.  Decompression scale factor of 8. | | 4 | sf16 | Compression scale factor of 1/16.  Decompression scale factor of 16. | | 5 | unsupported32 | This scale factor is unsupported and must not be used. | | 6 | unsupported64 | This scale factor is unsupported and must not be used. | | 7 | unsupported128 | This scale factor is unsupported and must not be used. | |
| 12 | sg\_ob\_charged | If this bit is set, then outbound scheduling group meters are to be charged.  If this bit is clear, then outbound scheduling group meters are not to be charged. |
| 11 | sg\_ib\_charged | If this bit is set, then inbound scheduling group meters are to be charged.  If this bit is clear, then inbound scheduling group meters are not to be charged. |
| 10 | qg\_ob\_charged | If this bit is set, then outbound queue group meters are to be charged.  If this bit is clear, then outbound queue group meters are not to be charged. |
| 9 | qg\_ib\_charged | If this bit is set, then inbound queue group meters are to be charged.  If this bit is clear, then inbound queue group meters are not to be charged. |
| 8:0 | queue\_group | This value represents the tqme queue group. |

#### vf (Offset:0xc[27:16], Size: 12)

VF for which the operation is for. This field is valid only if vf\_valid is '1'.

#### pf (Offset:0xc[31:28], Size: 4)

PF for which the operation is for. This field is always valid.

### AUX Command TLV

This TLV will be next in the stackup. This is the TLV provided from the host with the keys, IV, and GUID removed. The first beat is been modified to show the engine and frame and use BIP checking rather than a CRC.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Offset*** | ***31*** |  | | | | ***0*** |
| 0x0 | unused(4) | [eng](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#aux_cmd/NONE/NONE/eng) | [seq](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#aux_cmd/NONE/NONE/seq) | [length](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#aux_cmd/NONE/NONE/length) | | [type](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#aux_cmd/NONE/NONE/type) |
| 0x4 | [bip2](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#aux_cmd/NONE/NONE/bip2) | unused(19) | | | [frame](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#aux_cmd/NONE/NONE/frame) | |
| 0x8 | [frame\_ctrl](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#aux_cmd/NONE/NONE/frame_ctrl) | | | | | |
| 0xc | [debug\_ctrl](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#aux_cmd/NONE/NONE/debug_ctrl) | | | | | |
| 0x10 | [comp\_ctrl](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#aux_cmd/NONE/NONE/comp_ctrl) | | | | | |
| 0x14 | [crypto\_ctrl](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#aux_cmd/NONE/NONE/crypto_ctrl) | | | | | |

**Bits: 169 / Bytes: 22 / Unused Bits: 23**

#### type (Offset:0x0[7:0], Size: 8)

TLV Type - The value in this field defines the field format to apply to the entire TLV.

|  |  |  |
| --- | --- | --- |
| ***Value*** | ***Enum*** | ***Enumeration Description*** |
| 1 | cmd | Command TLV |

#### length (Offset:0x0[15:8], Size: 8)

Length of the TLV in units of 4 bytes. This length is computed over the entirety of the TLV Type, Length and data fields.

#### seq (Offset:0x0[23:16], Size: 8)

Per-Engine sequence number for the operation. This value will be incremented by one for each operation on an engine. All TLV for the same operation will have the same sequence number.

#### eng (Offset:0x0[27:24], Size: 4)

Engine ID for the operation.

#### frame (Offset:0x4[10:0], Size: 11)

Frame number within operation. This field will be zero for TLV that occur just once within the command.

#### bip2 (Offset:0x4[31:30], Size: 2)

2bit parity value over the first 64b of the TLV. bip2[1] contains xor computed over the odd bits in the first 64b word of TLV excluding bip2[1]. bip2[0] contains xor computed over the even bits in the first 64b word of TLV excluding bip2[0].

#### frame\_ctrl (Offset:0x8[31:0], Size: 32)

Frame Data Control

|  |  |  |
| --- | --- | --- |
| ***Offset*** | ***Field*** | ***Field Description*** |
| 31 | Unused | - |
| 30 | dst\_aux\_frame\_data\_guid\_present | Determines if GUID is present on the front of AUX\_FRAME\_DATA output.   |  |  |  | | --- | --- | --- | | ***Value*** | ***Enum*** | ***Enumeration Description*** | | 0 | NONE | No GUID is present at the front of the AUX\_FRAME\_DATA output. The AUX\_FRAME\_DATA output will start with the array of data as defined by the out\_type field. | | 1 | PRESENT | 256b GUID will be generated at the front of the AUX\_FRAME\_DATA output. | |
| 29:23 | out\_type | This value defines the format of the DST\_AUX\_FRAME\_DATA data after the GUID (if present).   |  |  |  | | --- | --- | --- | | ***Value*** | ***Enum*** | ***Enumeration Description*** | | 11 | NONE | NONE AUX\_FRAME\_DATA User Format | | 12 | NVME\_PI | 8B structure with PIB User Format | | 13 | PI64 | CRC64 Protection Index User Format | | 14 | VM\_USER | VM User Format | | 15 | APPEND | Append Intermediate Format | | 16 | IN\_PLACE\_SHORT | In-Place Write Short Intermediate Format | | 17 | IN\_PLACE\_LONG | In-Place Write Long Intermediate Format | | 18 | VM\_LONG\_IM | VM Long Intermediate Format | | 22 | VM\_SHORT\_IM | VM Short Intermediate Format | |
| 22:21 | md\_op | This field defines any operation done on the metadata area.   |  |  |  | | --- | --- | --- | | ***Value*** | ***Enum*** | ***Enumeration Description*** | | 0 | CRC | This performs the normal CRC metadata operation.  For compression/encryption this means that the CRC in the metadata will be verified.  For decryption/decompression this means that the CRC value will be generated. | |
| 20:19 | nvme\_md\_type | NVME MD format - A NVME MD block is included in some of the AUX\_FRAME\_DATA formats. If it is provided, then this field defines the format of the data within that field.   |  |  |  | | --- | --- | --- | | ***Value*** | ***Enum*** | ***Enumeration Description*** | | 0 | NOP\_8B | 8B structure with no CRC. | | 1 | CRC64\_8B | 8B structure with CRC64. | | 2 | CRC64E\_8B | 8B structure with CRC64E. | |
| 18:12 | src\_aux\_frame\_data\_format | This value defines the format of the SRC\_AUX\_FRAME\_DATA data after the GUID (if present).   |  |  |  | | --- | --- | --- | | ***Value*** | ***Enum*** | ***Enumeration Description*** | | 11 | NONE | NONE AUX\_FRAME\_DATA User Format | | 12 | NVME\_PI | 8B structure with PIB User Format | | 13 | PI64 | CRC64 Protection Index User Format | | 14 | VM\_USER | VM User Format | | 15 | APPEND | Append Intermediate Format | | 16 | IN\_PLACE\_SHORT | In-Place Write Short Intermediate Format | | 17 | IN\_PLACE\_LONG | In-Place Write Long Intermediate Format | | 18 | VM\_LONG\_IM | VM Long Intermediate Format | | 22 | VM\_SHORT\_IM | VM Short Intermediate Format | |
| 11:6 | src\_aux\_frame\_data\_size | Size of the input per-frame AUX data in 4 byte units. 0 to 252B. |
| 5 | aux\_frame\_data\_crc\_present | AUX\_FRAME\_DATA CRC Control   |  |  |  | | --- | --- | --- | | ***Value*** | ***Enum*** | ***Enumeration Description*** | | 0 | NONE | No CRC is at the end of the SRC\_AUX\_FRAME\_DATA buffer or the DST\_AUX\_FRAME\_DATA buffer. | | 1 | PRESENT | For decryption/decompression there is a CRC32 at the end of the SRC\_AUX\_FRAME\_DATA buffer that covers the entire SRC\_AUX\_FRAME\_DATA structure including the GUID, if present.  For compression/encryption there is a CRC32 at the end of the DST\_AUX\_FRAME\_DATA buffer that covers the entire DST\_AUX\_FRAME\_DATA structure including the GUID, if present.  CRC location is set to zeros in the calculation. | |
| 4 | src\_aux\_frame\_data\_guid\_present | Determines if GUID is present at front of input AUX\_FRAME\_DATA.   |  |  |  | | --- | --- | --- | | ***Value*** | ***Enum*** | ***Enumeration Description*** | | 0 | NONE | No GUID is present on the input AUX\_FRAME\_DATA. If AUX\_CMD format is 0 or 1 are used then no GUID is provided. If AUX\_CMD formtt is 2 or 3, then GUID is located in the AUX\_CMD. | | 1 | PRESENT | The GUID is provided on the input AUX\_FRAME\_DATA. If the AUX\_CMD format field is 2 or 3, the GUID in the AUX\_CMD will take precedence over the GUID provided in the AUX\_FRAME\_DATA. | |
| 3:0 | frame\_size | Frame size for command. This defines the frame size for the command.   |  |  |  | | --- | --- | --- | | ***Value*** | ***Enum*** | ***Enumeration Description*** | | 0 | SIMPLE | Command will be executed as a single frame up to 8MB. | | 5 | COMPOUND\_4K | Command will be broken up into 4KB frames. | | 6 | COMPOUND\_8K | Command will be broken up into 8KB frames. | |

#### debug\_ctrl (Offset:0xc[31:0], Size: 32)

Debug control fields. Set this register to zero for normal operation.

*Debug operations must be enabled for the device for this field to be active.*

|  |  |  |
| --- | --- | --- |
| ***Offset*** | ***Field*** | ***Field Description*** |
| 31 | dbg\_dest | Field to select destination of the debug command.   |  |  |  | | --- | --- | --- | | ***Value*** | ***Enum*** | ***Enumeration Description*** | | 0 | USER | Debug command targeted for USER logic. (This mode is only used by the CRC modules) | | 1 | TLVP | Debug command targeted for TLV parser. | |
| 30:29 | cmd\_mode | |  |  |  | | --- | --- | --- | | ***Value*** | ***Enum*** | ***Enumeration Description*** | | 0 | SINGLE | Signle error injection. | | 1 | CONT\_ENA | Start continuous error mode. | | 2 | CONT\_DIS | End continuous error mode. | |
| 28:24 | module\_id | |  |  |  | | --- | --- | --- | | ***Value*** | ***Enum*** | ***Enumeration Description*** | | 0 | IN\_STRM\_FIFO | Input Stream FIFO | | 1 | RAW\_CRC | Raw CRC Generator/Checker | | 2 | PRFIX\_ENG | Prefix Engine | | 3 | PRFIX\_ATTACH | Prefix Attach | | 4 | LZ77\_COMP | LZ77 Compressor | | 5 | HUFF\_ENC\_IN | Huffman Encoder Input Parser | | 6 | ENCRYPT\_ENG | Encrypt Engine | | 7 | CRC\_GEN | Compressed/Encrypted CRC Generator | | 8 | CRC\_CHK | Compressed/Encrypted CRC Checker | | 9 | DECRYPT\_ENG | Decrypt Engine | | 10 | HUFF\_DEC | Huffman Decoder | | 11 | AUTH\_ENG | Authentication Engine | | 12 | RAW\_CRC\_CHK | Raw CRC Checker | | 13 | CMPL\_GEN | Completion Generator | | 14 | OUT\_STRM\_FIFO | Output Stream FIFO | | 15 | STAT\_ACCUM | Statistics Accumulator | | 16 | HUFF\_ENC | Huffman Encoder Output Parser | |
| 23 | cmd\_type | Field to select debug operation of the TLV Parser only. Must be 0 for USER mode.   |  |  |  | | --- | --- | --- | | ***Value*** | ***Enum*** | ***Enumeration Description*** | | 0 | DATAPATH | Datapath corrupt. ( use this for XOR corruption). | | 1 | BACKPRESSURE | Create backpressure at the module inbound interface per settings in bits [17:0] | |
| 22:18 | tlv\_type | When cmd\_type=BACKPRESSURE, the tlv\_type field is unused.  For other modes, this is the TLV type to modify with the debug operation.   |  |  |  | | --- | --- | --- | | ***Value*** | ***Enum*** | ***Enumeration Description*** | | 0 | rqe | RQE TLV | | 1 | cmd | Command TLV | | 2 | key | Key TLV | | 5 | data | Data TLV | | 8 | stats | Statistics TLV | | 9 | cqe | CQE TLV | | 10 | guid | GUID TLV | | 11 | frmd\_user\_none | Aux Frame Data User None | | 12 | frmd\_user\_pi16 | Aux Frame Data User NVME PI16 | | 13 | framd\_user\_pi64 | Aux Frame Data User PI64 | | 14 | frmd\_user\_vm | Aux Frame Data User VM | | 15 | frmd\_int\_append | Aux Frame Data Intermediate Append | | 16 | frmd\_int\_in\_place\_short | Aux Frame Data Intermediate In-Place Write Short | | 17 | frmd\_int\_in\_place\_long | Aux Frame Data Intermediate In-Place Write Long | | 18 | frmd\_int\_vm\_long\_im | Aux Frame Data Intermediate VM Long | | 21 | AUX\_CMD | Command TLV - Used in AUX\_CMD to carry engine commands. This version has no GUID or IV/Tweak.  If crypto is enabled then then IV/Tweak is located within the AUX\_FRAME\_DATA structure. The GUID, if needed is located at the front of the AUX\_FRAME\_DATA.  This is the AUX\_CMD format that must be used if crypto is disabled for the operation. | | 22 | frmd\_int\_vm\_short\_im | Aux Frame Data Intermediate VM Short | | 23 | AUX\_CMD\_IV | Command TLV - Used in AUX\_CMD to carry engine commands. This version has no GUID but has an IV/Tweak.  If crypto is enabled and the GUID is needed, then the GUID is located at the front of the AUX\_FRAME\_DATA. | | 24 | AUX\_CMD\_GUID | Command TLV - Used in AUX\_CMD to carry engine commands. This version has a GUID but no IV/Tweak.  If crypto is enabled then the IV/Tweak is located within the AUX\_FRAME\_DATA structure. | | 25 | AUX\_CMD\_GUID\_IV | Command TLV - Used in AUX\_CMD to carry engine commands. This version includes both a GUID and a IV/Tweak. | | 26 | sched | Scheduler Update TLV | |
| 17:8 | byte\_num | When cmd\_type=BACKPRESSURE, the byte\_num field is defined as the backpressure assert count  For other modes, this value is the byte number to corrupt. Value can be 0 to 1023. |
| 7:0 | byte\_msk | When cmd\_type=BACKPRESSURE, the byte\_msk field is defined as the backpressure de-assert count  For other modes, this is the value that will be XOR with the datapath value to create the error injection. |

#### comp\_ctrl (Offset:0x10[31:0], Size: 32)

This register controls the compression/decompression operation.

|  |  |  |
| --- | --- | --- |
| ***Offset*** | ***Field*** | ***Field Description*** |
| 31:24 | Unused | - |
| 23:22 | comp\_threshold | Output size check on emitted frame. For ZLIB, GZIP, and XP9 this will be set to the frame size. For CHU modes can force a check of minus 16B. The last option is for debug or systems that do not want raw data.   |  |  |  | | --- | --- | --- | | ***Value*** | ***Enum*** | ***Enumeration Description*** | | 0 | FRAME\_SIZE | Frame Size used as compare for compressibility. | | 1 | FRAME\_SIZE\_SUB16 | Compressible threshold is CHU Frame size - 16. | | 2 | FORCE\_CODING | Force Coding Blocks - Do not emit raw data. | |
| 21 | xp10\_crc\_mode | This value controls the size of the CRC in the xp10 header.   |  |  |  | | --- | --- | --- | | ***Value*** | ***Enum*** | ***Enumeration Description*** | | 0 | B32CRC | CRC will be a 32b value | | 1 | B64CRC | CRC will be a 64b value | |
| 20:15 | xp10\_prefix\_selector | XP10\_prefix\_mode=0: This field is unused.  XP10\_prefix\_mode=2 or 3: This field selects one of 63 pre-defined prefixes to load. To use the Prefix Engine, SW should populate this field with zero, otherwise this value will be the predetermined prefix used.  XP10\_prefix\_mode=1: Define the size in 1KB blocks of the user defined prefix data at the beginning of the input data buffer. Prefix data will be followed by a CRC32C check value. Value of 0 indicates a 1KB prefix. Value of 1 indicates a 2KB prefix. Value of 63 indicates a 64KB prefix. |
| 14:13 | xp10\_prefix\_mode | XP10 Prefix Mode. Determines prefix processing mode of the CCEIP/CDDIP.   |  |  |  | | --- | --- | --- | | ***Value*** | ***Enum*** | ***Enumeration Description*** | | 0 | NONE | Normal mode. Data is compressed/decompressed normally, and no prefix data is present. | | 1 | USER\_DEFINED\_PREFIX | User defined prefix mode. Match Engine uses the incoming prefix data to setup subsequent matches. This is only supported for non-compound operations. Length of prefix data is determined by the xp10\_prefix\_selector field. | | 2 | PREFIX | Prefix mode. Match window is loaded with Prefix data selected by the xp10\_prefix\_selector field. | | 3 | PREFIX\_WITH\_HUFFMAN | Prefix with Huffman mode. As in PREFIX, except the Huffman Encoder is also loaded with Predetermined Huffman data selected by the xp10\_prefix\_selector field. (This mode is also referred to as pre-determined Huffman). | |
| 12:11 | lz77\_max\_symbol\_len | Maximum LZ77 Symbol Length   |  |  |  | | --- | --- | --- | | ***Value*** | ***Enum*** | ***Enumeration Description*** | | 0 | UNRESTRICTED | Unrestricted. Uses window size. | | 1 | BYTES\_258 | 258B (Deflate) | | 2 | MIN\_MATCH\_SIZE\_PLUS\_14 | Minimum Match Size + 14  This will force no LONG symbol Generation for the XP modes). | | 3 | BYTES\_64 | 64B (Debug) | |
| 10 | lz77\_min\_match\_len | Number of characters per n-gram.  Used by CCEIP only and ignored by CDDIP.   |  |  |  | | --- | --- | --- | | ***Value*** | ***Enum*** | ***Enumeration Description*** | | 0 | char\_3 | 3 characters per n-gram | | 1 | char\_4 | 4 characters per n-gram | |
| 9:8 | lz77\_delayed\_match\_window\_select | Compression Delayed Match Window Select. Ignored by CCDIP. Delays match selection by the given number of characters.  *It is expected that GZIP/ZLIB formats will set this field to one.*   |  |  |  | | --- | --- | --- | | ***Value*** | ***Enum*** | ***Enumeration Description*** | | 0 | NONE | No delayed match window | | 1 | CHAR\_1 | One character delayed match window | | 2 | CHAR\_2 | Two character delayed match window | |
| 7:4 | lz77\_window\_size | LZ77 Window Size   |  |  |  | | --- | --- | --- | | ***Value*** | ***Enum*** | ***Enumeration Description*** | | 0 | SIZE\_32B | 32B (For testing) | | 1 | SIZE\_4KB | 4KB | | 2 | SIZE\_8KB | 8KB | | 3 | SIZE\_16KB | 16KB | | 4 | SIZE\_32KB | 32KB | | 5 | SIZE\_64KB | 64KB (32K for ZLIB/GZIP) | |
| 3:0 | algorithm | Compression Algorithm   |  |  |  | | --- | --- | --- | | ***Value*** | ***Enum*** | ***Enumeration Description*** | | 0 | NONE | No compression. Data passed through engine. | | 1 | ZLIB | ZLIB Deflate (RFC-1950) | | 2 | GZIP | GZIP Deflate (RFC-1952) | | 3 | XPRESS9L6 | Xpress9L6 | | 4 | XPRESS10 | Xpress10 | | 5 | XPRESS10\_COMP\_4KB | Xpress10 Compact format with 4KB max frame size | | 6 | XPRESS10\_COMP\_8KB | Xpress10 Compact format with 8KB max frame size | |

#### crypto\_ctrl (Offset:0x14[31:0], Size: 32)

This field control the crypto operation for the command.

|  |  |  |
| --- | --- | --- |
| ***Offset*** | ***Field*** | ***Field Description*** |
| 31 | Unused | - |
| 30:25 | key\_type | This field controls which structure is used to desecribe the crypto keys.   |  |  |  | | --- | --- | --- | | ***Value*** | ***Enum*** | ***Enumeration Description*** | | 0 | NONE | No key information needed. This value is used when crypto is not being used. | |
| 24:23 | Unused | - |
| 22 | cipher\_pad\_op | Cipher padding option   |  |  |  | | --- | --- | --- | | ***Value*** | ***Enum*** | ***Enumeration Description*** | | 0 | NONE | No padding | | 1 | B16 | ANSI Padding to 16B boundary | |
| 21:20 | iv\_op | IV Operation   |  |  |  | | --- | --- | --- | | ***Value*** | ***Enum*** | ***Enumeration Description*** | | 0 | IV\_SRC | Directly from iv\_src location | | 1 | RANDOM | From Random number generator | | 2 | INCREMENT | Increment IV for each frame. | |
| 19:12 | aad\_len | Length of AAD from start of frame in Bytes (AES-GCM only) |
| 11:8 | cipher\_op | Cipher Operation Type   |  |  |  | | --- | --- | --- | | ***Value*** | ***Enum*** | ***Enumeration Description*** | | 0 | NONE | No Cipher operation done | | 1 | AES\_GCM | AES-GCM | | 2 | AES\_XTS | AES-XTS | | 3 | AES\_GMAC | AES-GMAC | |
| 7:4 | auth\_op | Authentication Operation Type   |  |  |  | | --- | --- | --- | | ***Value*** | ***Enum*** | ***Enumeration Description*** | | 0 | NONE | No Authentication done | | 1 | SHA2 | SHA2 | | 2 | HMAC\_SHA2 | HMAC-SHA2 | |
| 3:0 | raw\_auth\_op | Raw Data Authentication Operation Type   |  |  |  | | --- | --- | --- | | ***Value*** | ***Enum*** | ***Enumeration Description*** | | 0 | NONE | No Authentication done | | 1 | SHA2 | SHA2 | | 2 | HMAC\_SHA2 | HMAC-SHA2 | |

### AUX FRMD IN TLV

This is the structure that passes the AUX\_FRMD information. The header identifies what AUX\_FRMD format the data is in and the data follows the header.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***Offset*** | ***31*** |  | | | | ***0*** |
| 0x0 | unused(4) | [eng](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#frmd_in/NONE/NONE/eng) | [seq](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#frmd_in/NONE/NONE/seq) | [length](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#frmd_in/NONE/NONE/length) | | [type](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#frmd_in/NONE/NONE/type) |
| 0x4 | [bip2](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#frmd_in/NONE/NONE/bip2) | unused(19) | | | [frame](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#frmd_in/NONE/NONE/frame) | |
| 0x8  -  0x47 | [aux\_frmd[VAR]](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#frmd_in/NONE/NONE/aux_frmd) | | | | | |

**Bits: 553 / Bytes: 70 / Unused Bits: 23**

#### type (Offset:0x0[7:0], Size: 8)

TLV Type - The value in this field defines the field format to apply to the entire TLV.

|  |  |  |
| --- | --- | --- |
| ***Value*** | ***Enum*** | ***Enumeration Description*** |
| 11 | frmd\_user\_none | Aux Frame Data User None |
| 12 | frmd\_user\_pi16 | Aux Frame Data User NVME PI16 |
| 13 | framd\_user\_pi64 | Aux Frame Data User PI64 |
| 14 | frmd\_user\_vm | Aux Frame Data User VM |
| 15 | frmd\_int\_append | Aux Frame Data Intermediate Append |
| 16 | frmd\_int\_in\_place\_short | Aux Frame Data Intermediate In-Place Write Short |
| 17 | frmd\_int\_in\_place\_long | Aux Frame Data Intermediate In-Place Write Long |
| 18 | frmd\_int\_vm\_long\_im | Aux Frame Data Intermediate VM Long |
| 22 | frmd\_int\_vm\_short\_im | Aux Frame Data Intermediate VM Short |

#### length (Offset:0x0[15:8], Size: 8)

Length of the TLV in units of 4 bytes. This length is computed over the entirety of the TLV Type, Length and data fields.

The length may indicate that only part of the data in the last beat is valid. The next TLV will start at bit zero of the interface. Unused area in the last beat will be sent as zero.

#### seq (Offset:0x0[23:16], Size: 8)

Per-Engine sequence number for the operation. This value will be incremented by one for each operation on an engine. All TLV for the same operation will have the same sequence number.

#### eng (Offset:0x0[27:24], Size: 4)

Engine ID for the operation.

#### frame (Offset:0x4[10:0], Size: 11)

Frame number within operation. This field will be zero for TLV that occur just once within the command.

#### bip2 (Offset:0x4[31:30], Size: 2)

2bit parity value over the first 64b of the TLV. bip2[1] contains xor computed over the odd bits in the first 64b word of TLV excluding bip2[1]. bip2[0] contains xor computed over the even bits in the first 64b word of TLV excluding bip2[0].

#### aux\_frmd (Offset:0x8[31:0], Size: 32, Words: VAR)

This area is filled with the aux\_frmd format selected by the type field.

##### 8B structure with PIB User Format (12)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Offset*** | ***31*** |  | | ***0*** |
| 0x0 | [app\_tag](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#nvme_pi/NONE/NONE/app_tag) | | [guard](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#nvme_pi/NONE/NONE/guard) | |
| 0x4 | [ref\_tag](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#nvme_pi/NONE/NONE/ref_tag) | | | |

**Bits: 64 / Bytes: 8 / Unused Bits: 0**

###### guard (Offset:0x0[15:0], Size: 16)

16b DIX CRC guard value.

###### app\_tag (Offset:0x0[31:16], Size: 16)

Application Tag value for META DATA.

###### ref\_tag (Offset:0x4[31:0], Size: 32)

Reference Tag value for META DATA.

##### CRC64 Protection Index User Format (13)

|  |  |  |  |
| --- | --- | --- | --- |
| ***Offset*** | ***31*** |  | ***0*** |
| 0x0  -  0x7 | [guard](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#pi64/NONE/NONE/guard) | | |

**Bits: 64 / Bytes: 8 / Unused Bits: 0**

###### guard (Offset:0x0[31:0], Size: 32, Words: 2)

64b guard value.

##### VM User Format (14)

|  |  |  |  |
| --- | --- | --- | --- |
| ***Offset*** | ***31*** |  | ***0*** |
| 0x0  -  0xf | [iv\_tweak](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#vm_user/NONE/NONE/iv_tweak) | | |
| 0x10  -  0x17 | [raw\_data\_cksum](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#vm_user/NONE/NONE/raw_data_cksum) | | |
| 0x18  -  0x37 | [raw\_data\_mac](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#vm_user/NONE/NONE/raw_data_mac) | | |

**Bits: 448 / Bytes: 56 / Unused Bits: 0**

###### iv\_tweak (Offset:0x0[31:0], Size: 32, Words: 4)

IV (95:0) or Tweak (127:0) to be used for the encryption of the frame.

###### raw\_data\_cksum (Offset:0x10[31:0], Size: 32, Words: 2)

RAW Data Checksum. Is the CRC64 calculated over the raw data.

###### raw\_data\_mac (Offset:0x18[31:0], Size: 32, Words: 8)

RAW Data MAC. SHA2, or SHA2-HMAC over raw data.

##### Append Intermediate Format (15)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Offset*** | ***31*** |  | | ***0*** |
| 0x0  -  0x7 | [cipher\_data\_cksum](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#append/NONE/NONE/cipher_data_cksum) | | | |
| 0x8  -  0x17 | [cipher\_data\_mac](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#append/NONE/NONE/cipher_data_mac) | | | |
| 0x18  -  0x23 | [iv](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#append/NONE/NONE/iv) | | | |
| 0x24 | [reserved](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#append/NONE/NONE/reserved) | | | |
| 0x28 | [status](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#append/NONE/NONE/status) | | [data\_len](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#append/NONE/NONE/data_len) | |

**Bits: 352 / Bytes: 44 / Unused Bits: 0**

###### cipher\_data\_cksum (Offset:0x0[31:0], Size: 32, Words: 2)

CRC64 over the cipher data.

###### cipher\_data\_mac (Offset:0x8[31:0], Size: 32, Words: 4)

AES-GCM MAC over cipher data.

###### iv (Offset:0x18[31:0], Size: 32, Words: 3)

IV used to encrypt this frame.

###### reserved (Offset:0x24[31:0], Size: 32)

###### data\_len (Offset:0x28[23:0], Size: 24)

When data is compressed and not encrypted: this is the compressed data frame length (this is in bytes as opposed to "compressed data length" in bits, which is stored in some of the XP headers).

When data is compressed and encrypted: this is the cipher data length (which is the actual compressed data length including any AES padding).

When data is not compressed and not encrypted: this value is the actual data length.

When data is not compressed and encrypted: this value is the cipher data length (which is the actual data length including any AES padding).

###### status (Offset:0x28[31:24], Size: 8)

|  |  |  |
| --- | --- | --- |
| ***Offset*** | ***Field*** | ***Field Description*** |
| 7:2 | Unused | - |
| 1:0 | kind | The kind identifies the format for the intermediate data. This value is used to control decompression.   |  |  |  | | --- | --- | --- | | ***Value*** | ***Enum*** | ***Enumeration Description*** | | 0 | PARSEABLE | The field was compressed using XP9, XP10, Zlib or GZIP. The compressed data contains a header that should be used to determine how it was compressed. | | 1 | RAW | The field was not compressed. Data is RAW data. | | 2 | XP10\_COMPRESSED\_4K | The field was compressed using XP10 but has a compresssed header. The frame size was 4KB. | | 3 | XP10\_COMPRESSED\_8K | The field was compressed using XP10 but has a compressed header. The frame size was 8KB. | |

##### In-Place Write Short Intermediate Format (16)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Offset*** | ***31*** |  | | ***0*** |
| 0x0  -  0x7 | [cipher\_data\_cksum](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#in_place_short/NONE/NONE/cipher_data_cksum) | | | |
| 0x8  -  0xf | [raw\_data\_mac](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#in_place_short/NONE/NONE/raw_data_mac) | | | |
| 0x10 | [status](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#in_place_short/NONE/NONE/status) | | [data\_len](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#in_place_short/NONE/NONE/data_len) | |

**Bits: 160 / Bytes: 20 / Unused Bits: 0**

###### cipher\_data\_cksum (Offset:0x0[31:0], Size: 32, Words: 2)

CRC64 over the cipher data.

###### raw\_data\_mac (Offset:0x8[31:0], Size: 32, Words: 2)

Truncated RAW DATA MAC

###### data\_len (Offset:0x10[23:0], Size: 24)

When data is compressed and not encrypted: this is the compressed data frame length (this is in bytes as opposed to "compressed data length" in bits, which is stored in some of the XP headers).

When data is compressed and encrypted: this is the cipher data length (which is the actual compressed data length including any AES padding).

When data is not compressed and not encrypted: this value is the actual data length.

When data is not compressed and encrypted: this value is the cipher dasta length (which is the actual data length including any AES padding).

###### status (Offset:0x10[31:24], Size: 8)

|  |  |  |
| --- | --- | --- |
| ***Offset*** | ***Field*** | ***Field Description*** |
| 7:2 | Unused | - |
| 1:0 | kind | The kind identifies the format for the intermediate data. This value is used to control decompression.   |  |  |  | | --- | --- | --- | | ***Value*** | ***Enum*** | ***Enumeration Description*** | | 0 | PARSEABLE | The field was compressed using XP9, XP10, Zlib or GZIP. The compressed data contains a header that should be used to determine how it was compressed. | | 1 | RAW | The field was not compressed. Data is RAW data. | | 2 | XP10\_COMPRESSED\_4K | The field was compressed using XP10 but has a compresssed header. The frame size was 4KB. | | 3 | XP10\_COMPRESSED\_8K | The field was compressed using XP10 but has a compressed header. The frame size was 8KB. | |

##### In-Place Write Long Intermediate Format (17)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Offset*** | ***31*** |  | | ***0*** |
| 0x0  -  0x7 | [cipher\_data\_cksum](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#in_place_long/NONE/NONE/cipher_data_cksum) | | | |
| 0x8  -  0x27 | [raw\_data\_mac](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#in_place_long/NONE/NONE/raw_data_mac) | | | |
| 0x28 | [status](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#in_place_long/NONE/NONE/status) | | [data\_len](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#in_place_long/NONE/NONE/data_len) | |

**Bits: 352 / Bytes: 44 / Unused Bits: 0**

###### cipher\_data\_cksum (Offset:0x0[31:0], Size: 32, Words: 2)

CRC64 over the cipher data.

###### raw\_data\_mac (Offset:0x8[31:0], Size: 32, Words: 8)

RAW Data MAC

###### data\_len (Offset:0x28[23:0], Size: 24)

When data is compressed and not encrypted: this is the compressed data frame length (this is in bytes as opposed to "compressed data length" in bits, which is stored in some of the XP headers).

When data is compressed and encrypted: this is the cipher data length (which is the actual compressed data length including any AES padding).

When data is not compressed and not encrypted: this value is the actual data length.

When data is not compressed and encrypted: this value is the cipher dasta length (which is the actual data length including any AES padding).

###### status (Offset:0x28[31:24], Size: 8)

|  |  |  |
| --- | --- | --- |
| ***Offset*** | ***Field*** | ***Field Description*** |
| 7:2 | Unused | - |
| 1:0 | kind | The kind identifies the format for the intermediate data. This value is used to control decompression.   |  |  |  | | --- | --- | --- | | ***Value*** | ***Enum*** | ***Enumeration Description*** | | 0 | PARSEABLE | The field was compressed using XP9, XP10, Zlib or GZIP. The compressed data contains a header that should be used to determine how it was compressed. | | 1 | RAW | The field was not compressed. Data is RAW data. | | 2 | XP10\_COMPRESSED\_4K | The field was compressed using XP10 but has a compresssed header. The frame size was 4KB. | | 3 | XP10\_COMPRESSED\_8K | The field was compressed using XP10 but has a compressed header. The frame size was 8KB. | |

##### VM Long Intermediate Format (18)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Offset*** | ***31*** |  | | ***0*** |
| 0x0  -  0x7 | [cipher\_data\_cksum](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#vm_long_im/NONE/NONE/cipher_data_cksum) | | | |
| 0x8  -  0x27 | [cipher\_data\_mac](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#vm_long_im/NONE/NONE/cipher_data_mac) | | | |
| 0x28  -  0x47 | [raw\_data\_mac](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#vm_long_im/NONE/NONE/raw_data_mac) | | | |
| 0x48  -  0x57 | [iv\_tweak](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#vm_long_im/NONE/NONE/iv_tweak) | | | |
| 0x58 | [status](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#vm_long_im/NONE/NONE/status) | | [data\_len](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#vm_long_im/NONE/NONE/data_len) | |

**Bits: 736 / Bytes: 92 / Unused Bits: 0**

###### cipher\_data\_cksum (Offset:0x0[31:0], Size: 32, Words: 2)

CRC64 over the cipher data.

###### cipher\_data\_mac (Offset:0x8[31:0], Size: 32, Words: 8)

Cipher Data CKMICMAC (SHA2-HMAC, AES-GCM, or SHA2)

###### raw\_data\_mac (Offset:0x28[31:0], Size: 32, Words: 8)

Raw Data CKMICMAC (SHA2 or SHA2-HMAC)

###### iv\_tweak (Offset:0x48[31:0], Size: 32, Words: 4)

IV (95:0) or Tweak (127:0) to be used for the encryption of the frame.

###### data\_len (Offset:0x58[23:0], Size: 24)

When data is compressed and not encrypted: this is the compressed data frame length (this is in bytes as opposed to "compressed data length" in bits, which is stored in some of the XP headers).

When data is compressed and encrypted: this is the cipher data length (which is the actual compressed data length including any AES padding).

When data is not compressed and not encrypted: this value is the actual data length.

When data is not compressed and encrypted: this value is the cipher dasta length (which is the actual data length including any AES padding).

###### status (Offset:0x58[31:24], Size: 8)

|  |  |  |
| --- | --- | --- |
| ***Offset*** | ***Field*** | ***Field Description*** |
| 7:2 | Unused | - |
| 1:0 | kind | The kind identifies the format for the intermediate data. This value is used to control decompression.   |  |  |  | | --- | --- | --- | | ***Value*** | ***Enum*** | ***Enumeration Description*** | | 0 | PARSEABLE | The field was compressed using XP9, XP10, Zlib or GZIP. The compressed data contains a header that should be used to determine how it was compressed. | | 1 | RAW | The field was not compressed. Data is RAW data. | | 2 | XP10\_COMPRESSED\_4K | The field was compressed using XP10 but has a compresssed header. The frame size was 4KB. | | 3 | XP10\_COMPRESSED\_8K | The field was compressed using XP10 but has a compressed header. The frame size was 8KB. | |

##### VM Short Intermediate Format (22)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Offset*** | ***31*** |  | | ***0*** |
| 0x0  -  0x7 | [cipher\_data\_cksum](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#vm_short_im/NONE/NONE/cipher_data_cksum) | | | |
| 0x8  -  0x27 | [cipher\_data\_mac](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#vm_short_im/NONE/NONE/cipher_data_mac) | | | |
| 0x28  -  0x2f | [raw\_data\_mac](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#vm_short_im/NONE/NONE/raw_data_mac) | | | |
| 0x30  -  0x3f | [iv\_tweak](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#vm_short_im/NONE/NONE/iv_tweak) | | | |
| 0x40 | [status](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#vm_short_im/NONE/NONE/status) | | [data\_len](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#vm_short_im/NONE/NONE/data_len) | |

**Bits: 544 / Bytes: 68 / Unused Bits: 0**

###### cipher\_data\_cksum (Offset:0x0[31:0], Size: 32, Words: 2)

CRC64 over the cipher data.

###### cipher\_data\_mac (Offset:0x8[31:0], Size: 32, Words: 8)

Cipher Data CKMICMAC (SHA2-HMAC, AES-GCM, or SHA2)

###### raw\_data\_mac (Offset:0x28[31:0], Size: 32, Words: 2)

Truncated Raw Data CKMICMAC (SHA2 or SHA2-HMAC)

###### iv\_tweak (Offset:0x30[31:0], Size: 32, Words: 4)

IV (95:0) or Tweak (127:0) to be used for the encryption of the frame.

###### data\_len (Offset:0x40[23:0], Size: 24)

When data is compressed and not encrypted: this is the compressed data frame length (this is in bytes as opposed to "compressed data length" in bits, which is stored in some of the XP headers).

When data is compressed and encrypted: this is the cipher data length (which is the actual compressed data length including any AES padding).

When data is not compressed and not encrypted: this value is the actual data length.

When data is not compressed and encrypted: this value is the cipher data length (which is the actual data length including any AES padding).

###### status (Offset:0x40[31:24], Size: 8)

|  |  |  |
| --- | --- | --- |
| ***Offset*** | ***Field*** | ***Field Description*** |
| 7:2 | Unused | - |
| 1:0 | kind | The kind identifies the format for the intermediate data. This value is used to control decompression.   |  |  |  | | --- | --- | --- | | ***Value*** | ***Enum*** | ***Enumeration Description*** | | 0 | PARSEABLE | The field was compressed using XP9, XP10, Zlib or GZIP. The compressed data contains a header that should be used to determine how it was compressed. | | 1 | RAW | The field was not compressed. Data is RAW data. | | 2 | XP10\_COMPRESSED\_4K | The field was compressed using XP10 but has a compresssed header. The frame size was 4KB. | | 3 | XP10\_COMPRESSED\_8K | The field was compressed using XP10 but has a compressed header. The frame size was 8KB. | |

### Data TLV

This is the data for the frame.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ***Offset*** | ***31*** |  | | | | | ***0*** |
| 0x0 | unused(4) | | [eng](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#data_tlv/NONE/NONE/eng) | [seq](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#data_tlv/NONE/NONE/seq) | [length](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#data_tlv/NONE/NONE/length) | | [type](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#data_tlv/NONE/NONE/type) |
| 0x4 | [bip2](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#data_tlv/NONE/NONE/bip2) | [last\_of\_command](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#data_tlv/NONE/NONE/last_of_command) | unused(18) | | | [frame](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#data_tlv/NONE/NONE/frame) | |
| 0x8  -  0x47 | [data[VAR]](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#data_tlv/NONE/NONE/data) | | | | | | |

**Bits: 554 / Bytes: 70 / Unused Bits: 22**

#### type (Offset:0x0[7:0], Size: 8)

TLV Type - The value in this field defines the field format to apply to the entire TLV.

|  |  |  |
| --- | --- | --- |
| ***Value*** | ***Enum*** | ***Enumeration Description*** |
| 5 | data | Data TLV |

#### length (Offset:0x0[15:8], Size: 8)

The length value will be zero for the data TLV. The length of the data TLV will be signaled by the tuser signals. Not all bytes in the last beat of data need to be valid. Number of valid bytes is also indicated by the tuser signals.

#### seq (Offset:0x0[23:16], Size: 8)

Per-Engine sequence number for the operation. This value will be incremented by one for each operation on an engine. All TLV for the same operation will have the same sequence number.

#### eng (Offset:0x0[27:24], Size: 4)

Engine ID for the operation.

#### frame (Offset:0x4[10:0], Size: 11)

Frame number within operation. This field will be zero for TLV that occur just once within the command.

#### last\_of\_command (Offset:0x4[29], Size: 1)

This flag is zero for all data\_tlv except the last one in the command.

#### bip2 (Offset:0x4[31:30], Size: 2)

2bit parity value over the first 64b of the TLV. bip2[1] contains xor computed over the odd bits in the first 64b word of TLV excluding bip2[1]. bip2[0] contains xor computed over the even bits in the first 64b word of TLV excluding bip2[0].

#### data (Offset:0x8[31:0], Size: 32, Words: VAR)

Data body of the DATA TLV. This consists of the data to be processed for the frame.

### CQE TLV

This is the structure that communicates final status for an operation. This data will be forwarded to the CQE that is written to the host.

The tlast signal will be asserted for the last beat of this TLV indicating the end of all the data for a particular command.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ***Offset*** | ***31*** |  | | | | | ***0*** |
| 0x0 | unused(4) | [eng](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#cqe_tlv/NONE/NONE/eng) | [seq](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#cqe_tlv/NONE/NONE/seq) | | | [length](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#cqe_tlv/NONE/NONE/length) | [type](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#cqe_tlv/NONE/NONE/type) |
| 0x4 | [bip2](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#cqe_tlv/NONE/NONE/bip2) | unused(19) | | | | | [frame](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#cqe_tlv/NONE/NONE/frame) |
| 0x8 | [status\_code\_type](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#cqe_tlv/NONE/NONE/status_code_type) | [stat\_ctx](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#cqe_tlv/NONE/NONE/stat_ctx) | | | | [pf](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#cqe_tlv/NONE/NONE/pf) | [vf](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#cqe_tlv/NONE/NONE/vf) |
| 0xc | [status\_code](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#cqe_tlv/NONE/NONE/status_code) | | [do\_not\_resend](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#cqe_tlv/NONE/NONE/do_not_resend) | [vf\_valid](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#cqe_tlv/NONE/NONE/vf_valid) | unused(2) | [engine\_error\_code](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#cqe_tlv/NONE/NONE/engine_error_code) | [error\_frame](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_txc_eng.htm#cqe_tlv/NONE/NONE/error_frame) |

**Bits: 103 / Bytes: 13 / Unused Bits: 25**

#### type (Offset:0x0[7:0], Size: 8)

TLV Type - The value in this field defines the field format to apply to the entire TLV.

|  |  |  |
| --- | --- | --- |
| ***Value*** | ***Enum*** | ***Enumeration Description*** |
| 9 | cqe | CQE TLV |

#### length (Offset:0x0[15:8], Size: 8)

Length of the TLV in units of 4 bytes. This length is computed over the entirety of the TLV Type, Length and data fields.

The length may indicate that only part of the data in the last beat is valid. The next TLV will start at bit zero of the interface. Unused area in the last beat will be sent as zero.

#### seq (Offset:0x0[23:16], Size: 8)

Per-Engine sequence number for the operation. This value will be incremented by one for each operation on an engine. All TLV for the same operation will have the same sequence number.

#### eng (Offset:0x0[27:24], Size: 4)

Engine ID for the operation.

#### frame (Offset:0x4[10:0], Size: 11)

Frame number within operation. This field will be zero for TLV that occur just once within the command.

#### bip2 (Offset:0x4[31:30], Size: 2)

2bit parity value over the first 64b of the TLV. bip2[1] contains xor computed over the odd bits in the first 64b word of TLV excluding bip2[1]. bip2[0] contains xor computed over the even bits in the first 64b word of TLV excluding bip2[0].

#### vf (Offset:0x8[11:0], Size: 12)

VF for which the operation is for. This field is valid only if vf\_valid is '1'.

#### pf (Offset:0x8[12], Size: 1)

PF for which the operation is for. This field is always valid.

#### stat\_ctx (Offset:0x8[28:13], Size: 16)

The statistics context to aggregate this operation into.

#### status\_code\_type (Offset:0x8[31:29], Size: 3)

Status Code Type.

|  |  |  |
| --- | --- | --- |
| ***Value*** | ***Enum*** | ***Enumeration Description*** |
| 0 | GENERIC | Generic Command Status: Indicates that the command specified by the Command and SQ identifiers in the completion queue entry has completed. |

#### error\_frame (Offset:0xc[11:0], Size: 12)

If the status\_code is non-zero, this field gives the frame on which the error was discovered. If the error referred to something that was not frame oriented, this value will be all 1's.

#### engine\_error\_code (Offset:0xc[19:12], Size: 8)

Extended engine error code from the engine. This value is valid if the status code is 0xc7.

|  |  |  |
| --- | --- | --- |
| ***Value*** | ***Enum*** | ***Enumeration Description*** |
| 0 | NO\_ENGINE\_ERRORS | No Errors detected during Compression/Encryption Command |
| 1 | HD\_MEM\_ECC | Decoder ECC Failure |
| 2 | HD\_FHP\_PFX\_CRC | Decoder Predetermined Prefix CRC Mismatch |
| 3 | HD\_FHP\_PFX\_DATA\_ABSENT | Decoder Expected Prefix and didn't receive one |
| 4 | HD\_FHP\_PHD\_CRC | Decoder Predetermine Huffman Tables CRC Mismatch |
| 5 | HD\_FHP\_BAD\_FORMAT | Decoder Bad in Format Bad |
| 6 | HD\_BHP\_INVALID\_WSIZE | Decoder Received invalid window size in the Frame header for XP10 |
| 7 | HD\_BHP\_BLK\_CRC | Decoder XPx BLK CRC failure |
| 8 | HD\_BHP\_HDR\_INVALID | Decoder Received Invalid XPx Header (bits not 0) |
| 9 | HD\_BHP\_XP9\_HDR\_SEQ | Decoder XP9 Hdr Sequencer number not 0 |
| 10 | HD\_BHP\_XP10\_XTRA\_FLAG\_PRSNT | Decoder Received XP10 with XTRA FLAG Set (not supported) |
| 11 | HD\_BHP\_ZLIB\_FDICT\_PRSNT | Decoder ZLIB FDICT indicates Present (not supported) |
| 12 | HD\_BHP\_GZ\_CM\_NOT\_DEFLATE | Decoder GZIP header indicates deflate wasn't used as compression algorithm |
| 13 | HD\_BHP\_ZLIB\_CINFO\_RANGE | Decoder CINFO Out of Range |
| 14 | HD\_BHP\_ZLIB\_FCHECK | Decoder FCHECK check failed for ZLIB frame. |
| 15 | HD\_BHP\_DFLATE\_LEN\_CHECK | Decode LEN and NLEN check failed for RAW deflate block. |
| 16 | HD\_LFA\_REWIND\_FAIL | Deocoder Internal Error during Rewind - invalide Huffman Symbols |
| 17 | HD\_LFA\_PREMATURE\_EOF | Decoder Premature in Eof Premature |
| 18 | HD\_LFA\_LATE\_EOF | Decoder Late in Eof Late |
| 19 | HD\_LFA\_MISSING\_EOF | Decoder EOF not received at the end of last block |
| 20 | HD\_HTF\_XP9\_RESERVED\_SYMBOL\_TABLE\_ENCODING | Decoder received invalid XP9 symbol table encoding |
| 21 | HD\_HTF\_XP10\_RESERVED\_SYMBOL\_TABLE\_ENCODING | Decoder received invalid XP10 symbol table encoding |
| 22 | HD\_HTF\_XP10\_PREDEF\_SYMBOL\_TABLE\_ENCODING | Decoder received invalid XP10 predetermined symbol table encoding |
| 23 | HD\_HTF\_XP9\_ILLEGAL\_NONZERO\_BL | Decoder received an illegal non-zero bit length |
| 24 | HD\_HTF\_RLE\_OVERRUN | Decoder encounter a run length encoding overrun condition |
| 25 | HD\_HTF\_BAD\_HUFFMAN\_CODE | Decoder Bad in Code Huffman |
| 26 | HD\_HTF\_ILLEGAL\_SMALL\_HUFTREE | The block's unpacked short symbol table contains errors |
| 27 | HD\_HTF\_ILLEGAL\_HUFTREE | The block's unpacked long symbol table contains errors |
| 28 | HD\_HTF\_HDR\_UNDERRUN | Huffman header length is less than specified |
| 29 | HD\_BHP\_STBL\_SIZE\_ERR | Decoder symbol table size in the header does not match actual size. |
| 32 | HD\_SDD\_INVALID\_SYMBOL | Decoder Invalid in Symbol Invalid |
| 33 | HD\_SDD\_END\_MISMATCH | Decoder End in Mismatch End |
| 34 | HD\_SDD\_MISSING\_EOB\_SYM | Decoder Missing in Sym Eob |
| 35 | HD\_MTF\_XP9\_MTF3\_AFTER\_BACKREF | Decoder Xp9 Mtf3 in Backref After |
| 36 | HD\_MTF\_XP10\_MISSING\_MTF | Decoder Xp10 in Mtf Missing |
| 37 | HD\_BHP\_ILLEGAL\_MTF\_SZ | Decoder received MTF offsets greater than 16 bits |
| 38 | HD\_LZ\_HBIF\_SOFT\_OFLOW | Decompressor Fifo Overflow |
| 39 | HD\_BE\_FRM\_CRC | Decompressor CRC failed on Decompression (ADLER/XP9) |
| 40 | HD\_BE\_OLIMIT | Decompressor expanded beyond 8MB |
| 41 | HD\_BE\_SZ\_MISMATCH | Decomprssor expanded size doesn't match header expected size |
| 50 | CRCG\_CRC\_ERROR | Raw CRC Checker Error |
| 51 | CRCC0\_CRC\_ERROR | The CCEIP engine detected a CRC error that was calculated over the encrypted and compressed data, as output by the compressor. |
| 52 | CRCC1\_CRC\_ERROR | The CCEIP engine detected a CRC error that was calculated over the raw data, as output by the validator. |
| 54 | CRCGC0\_CRC\_ERROR | The CCEIP engine detected a CRC error that was calculated over the raw data, as input to the compressor. |
| 56 | CRCDC0\_CRC\_ERROR | The CDDIP engine detected a CRC error that was calculated over the encrypted and compressed data, as input to the decompressor. |
| 64 | LZ77\_COMP\_PREFIX\_CRC\_ERROR | Prefix CRC Check failed during LZ77 Compressor loading. |
| 65 | LZ77\_COMP\_INVALID\_COMP\_ALG | AUX\_CMD Compression Algorithm undefined code point |
| 66 | LZ77\_COMP\_INVALID\_WIN\_SIZE | AUX\_CMD Compression Algorithm Window Size undefined code point |
| 67 | LZ77\_COMP\_INVALID\_MIN\_LEN | AUX\_CMD Compression Algorithm Minimum Match Size undefined code point |
| 68 | LZ77\_COMP\_INVALID\_NUM\_MTF | Compression Algorithm Number of MTF undefined code point |
| 69 | LZ77\_COMP\_INVALID\_MAX\_LEN | Compression Algorithm Maximum Match Length undefined code point |
| 70 | LZ77\_COMP\_INVALID\_DMW\_SIZE | Compression Algorithm Delayed Match Window undefined code point |
| 71 | LZ77\_COMP\_LZ\_ERROR | LZ77 match state machine error detected |
| 80 | HE\_MEM\_ECC | Encoder Internal ECC Error |
| 81 | HE\_PDH\_CRC | Encoder CRC Check for Prefix Data Failed |
| 82 | HE\_PFX\_CRC | Encoder CRC Check for Predetermined Huffman Tables Failed |
| 83 | HE\_SYM\_MAP\_ERR | Encoder encountered an unmappable symbol |
| 100 | CRYPTO\_ENC\_SEED\_EXPIRED | Engine Seed Expired for DRNG |
| 101 | CRYPTO\_ENC\_IV\_MISSING | Engine Misconfigured - missing IV |
| 102 | CRYPTO\_ENC\_INVALID\_SEQNUM | Engine KEY or COMMAND Sequence Number Failed(didn't increment by +1) |
| 103 | CRYPTO\_ENC\_INVALID\_ENGINE\_ID | Engine KEY and COMMAND Engine ID Mismatch |
| 104 | CRYPTO\_ENC\_KEY\_TLV\_CRC\_ERROR | Key TLV CRC-32 mismatch detected |
| 105 | CRYPTO\_ENC\_MAL\_CMD | Invalid combinations of Crypto CMD TLV Word 2 |
| 106 | CRYPTO\_ENC\_XTS\_LEN\_ERROR | Data must be at least 16 bytes for a XTS operation |
| 107 | CRYPTO\_ENC\_AAD\_LEN\_ERROR | AAD Length should be strictly less than Data Length |
| 110 | CRYPTO\_DEC\_TAG\_MISCOMPARE | Engine Authentication Tag failed check |
| 112 | CRYPTO\_DEC\_IV\_MISSING | Engine Misconfigured - missing IV |
| 113 | CRYPTO\_DEC\_INVALID\_SEQNUM | Engine KEY or COMMAND Sequence Number Failed(didn't increment by +1) |
| 114 | CRYPTO\_DEC\_INVALID\_ENGINE\_ID | Engine KEY and COMMAND Engine ID Mismatch |
| 115 | CRYPTO\_DEC\_KEY\_TLV\_CRC\_ERROR | Key TLV CRC-32 mismatch detected |
| 116 | CRYPTO\_DEC\_MAL\_CMD | Invalid combinations of Crypto CMD TLV Word 2 |
| 117 | CRYPTO\_DEC\_XTS\_LEN\_ERROR | Data must be at least 16 bytes for a XTS operation |
| 118 | CRYPTO\_DEC\_AAD\_LEN\_ERROR | AAD Length should be strictly less than Data Length |
| 120 | CRYPTO\_INT\_TAG\_MISCOMPARE | Engine Authentication Tag failed check |
| 121 | CRYPTO\_INT\_INVALID\_SEQNUM | Engine Misconfigured - missing IV |
| 122 | CRYPTO\_INT\_INVALID\_ENGINE\_ID | Engine KEY or COMMAND Sequence Number Failed(didn't increment by +1) |
| 123 | CRYPTO\_INT\_KEY\_TLV\_CRC\_ERROR | Key TLV CRC-32 mismatch detected |
| 130 | KME\_DAK\_INV\_KIM | KIM Entry requested in AUX\_CMD for DAK |
| 131 | KME\_DAK\_PF\_VF\_VAL\_ERR | Validation Check failed DAK for this command |
| 132 | KME\_DEK\_INV\_KIM | KIM Entry requested in AUX\_CMD for DEK |
| 133 | KME\_DEK\_PF\_VF\_VAL\_ERR | Validation Check failed DEK for this command |
| 134 | KME\_SEED\_EXPIRED | Seed Expired for KDF |
| 135 | KME\_DEK\_GCM\_TAG\_FAIL | GCM Authentication Tag failed during DEK decryption |
| 136 | KME\_DAK\_GCM\_TAG\_FAIL | GCM Authentication Tag failed during DAK decryption |
| 137 | KME\_ECC\_FAIL | ECC failure on KIM/CKV data. |
| 138 | KME\_UNENC\_KEYS\_DISABLED | Key Type 2-6 is disabled. |
| 139 | KME\_DEK\_ILLEGAL\_KEK\_USAGE | KIM marked as KEK was neither used to decrypt DEK keys nor as a DEK KDF Key. |
| 140 | KME\_DAK\_ILLEGAL\_KEK\_USAGE | KIM marked as KEK was neither used to decrypt DAK keys nor as a DAK KDF Key. |
| 150 | PREFIX\_PC\_OVERRUN\_ERROR | Prefix Program Counter Overran (illegal program construction) |
| 151 | PREFIX\_NUM\_WR\_ERROR | Prefix Program Illegal Destination Write (Illegal program construction) |
| 152 | PREFIX\_ILLEGAL\_OPCODE | Prefix Program Illegal Opcode (Illegal program construction) |
| 153 | PREFIX\_ILLEGAL\_INST | Prefix Program Illegal Instruction (Opcode not correct) |
| 155 | PREFIX\_ATTACH\_PHD\_CRC\_ERROR | CRC for Selected Predetermined Huffman Table check failed |
| 156 | PREFIX\_ATTACH\_PFD\_CRC\_ERROR | CRC for Selected Predetermined Data check failed |
| 170 | CG\_UNDEF\_FRMD\_OUT | An undefined FRMD type was received |
| 180 | ISF\_PREFIX\_ERR | The ISF detected an invalid user prefix size |
| 255 | TLVP\_BIP2\_ERROR | Parser encountered a BIP2 failure in TLV header word |

#### vf\_valid (Offset:0xc[22], Size: 1)

When this bit is '1', the vf field is valid and the operation is for a VF. When this bit is zero, the operation is for a PF.

#### do\_not\_resend (Offset:0xc[23], Size: 1)

If set to '1', indicates that the same command if re-submitted is expected to fail.

#### status\_code (Offset:0xc[31:24], Size: 8)

Status Code - This field indicates the status for the CQE.

|  |  |  |
| --- | --- | --- |
| ***Value*** | ***Enum*** | ***Enumeration Description*** |
| 0 | SUCCESS | Successful Completion: The command completed successfully. |
| 1 | INVALID\_OP | Invalid command opcode. (TWE) |
| 2 | INVALID\_FIELD | Invalid or unsupported field specified in the command parameters. (TWE or RWE) |
| 4 | DATA | Transferring the data or AUX data associated with a command had an error. (TDC or RDIR) |
| 6 | INTERNAL | The command was not completed successfully due to an internal error. |
| 13 | INVALID\_SGL\_SEG\_DESCR | Invalid SGL Segment Descriptor: The command includes an invalid SGL Last Segment or SGL Segment descriptor. This may occur when the SGL segment pointed to by an SGL Last Segment descriptor contains an SGL Segment descriptor or an SGL Last Segment descriptor. This may occur when an SGL Segment descriptor or an SGL Last Segment descriptor contains an invalid length (i.e., a length of zero or one that is not a multiple of 16). (TWE or RWE) |
| 14 | INVALID\_NUM\_SGL\_DESCR | Invalid Number of SGL Descriptors: There is an SGL Last Segment descriptor or an SGL Segment descriptor in a location other than the last descriptor of a segment based on the length indicated. This may also occur when the last SGL in an SGL Segment (other than the last segment) is an SGL Data descriptor rather than an SGL Segment descriptor or an SGL Last Segment descriptor. (TWE or RWE) |
| 15 | INVALID\_DATA\_SGL\_LEN | Data SGL Length Invalid: This may occur if the length of a Data SGL is too short. This may occur if the length of a Data SGL is too long and the controller does not support SGL transfers longer than the amount of data to be transferred as indicated in the SGL Support field of the Identify Controller data structure. (TWE) |
| 16 | INVALID\_META\_SGL\_LEN | Metadata SGL Length Invalid: This may occur if the length of a Metadata SGL is too short. This may occur if the length of a Metadata SGL is too long and the controller does not support SGL transfers longer than the amount of data to be transferred as indicated in the SGL Support field of the Identify Controller data structure. (TWE) |
| 17 | INVALID\_SGL\_DESCR | SGL Descriptor Type Invalid: The type of an SGL Descriptor is a type that is not supported by the controller. (TWE or RWE) |
| 19 | INVALID\_PRP\_OFFSET | PRP Offset Invalid: The Offset field for a PRP entry is invalid. This may occur when there is a PRP entry with a non-zero offset after the first entry (not including the second entry if it is a PRP List Pointer). This may also occur when the second entry is a PRP List Pointer but the offset is not aligned to an 8B boundary in host memory. (TWE or RWE) |
| 22 | INVALID\_SGL\_OFFSET | SGL Offset Invalid: The offset specified in a descriptor is invalid. This may occur when there is an SGL Segment descriptor or an SGL Last Segment descriptor and the address is not aligned to an 8B boundary in host memory. (TWE or RWE) |
| 23 | INVALID\_SGL\_SUBTYPE | SGL Sub Type Invalid: The SGL Sub Type field specified is invalid. (TWE or RWE) |
| 192 | DATA\_BUFFER\_OVERRUN | This error is generated when the output data (either data or AUX\_FRAME\_DATA) exceeded the space provided in the RQE. (RWE) |
| 193 | FRAME\_SIZE\_OVERRUN | This error is generated when the output data for any frame exceeds the frame size for the operation as defined in the AUX\_CMDx and the frame\_size\_overrun\_en bit in the RQE is enabled. (RWE) |
| 194 | SGL\_OVER\_LIMIT | This error is generated when the aggregated length of any group of eight adjacent SGL list entries do not meet or exceed the data movement packet size. This applies to both data and AUX streams. This applies to all SGE types including an SGE pointer to additional SGEs. (TWE or RWE)  The "data movement packet" size is configured to 512B, 1K, 2K, 4K, or 8K and is globally configurable for data or aux\_data movement on the TX or RX sides of the chip. The nominal "data movement packet" size is 2KB for data and 512B for aux\_data.  Note that the purpose of this error is to detect and abort infinite pointer loops. Furthermore, as long as the rules for SGE assembly are followed, false errors should never be detected by hardware. However there are cases such that an SGE assembly rule violation may go undetected, but never such that an infinite loop will go un-noticed. |
| 195 | INVALID\_PHYS\_POINTER | Physical Pointer Type Invalid. The type of a Physical Pointer is a type that is not supported by the controller. (TWE or RWE) |
| 196 | DATA\_POINTER\_PCI\_ERROR | A PCI error was received when reading a data or AUX data pointer (SGL, PRP, or BEPL). (TWE or RWE) |
| 197 | AUX\_CMD\_MISMATCH | This error indicates that the AUX\_CMD area was either too big or two small for the TLV data inside the AUX\_CMD area. This means that TLV parse did not end at the end of the DMA size provided. (TXC) |
| 198 | AUX\_FRAME\_DATA\_MISMATCH | This error indicates that the AUX\_FRAME\_DATA area was either too big or too small for the data described by the AUX\_CMD. This means that TXC either encountered the end of the AUX\_FRAME\_DATA before exhausting the data or there was AUX\_FRAME\_DATA left over after all the data had been processed. (TXC) |
| 199 | ENGINE\_ERR | This status code indicates that the engine error code value is non-zero. Consult the engine error code for more details about the failure. |
| 200 | MAX\_DATA\_EXCEEDED | This status code will be returned if the source or destination data lengths specified in the RQE ever exceed the 8MB supported size limit, regardless of if the operation is simple or compound. (TWE/RWE) |
| 201 | SGL\_PTR\_TO\_PTR | This error is generated when an SGL of type segment descriptor points to another SGL of type segment (or last segment) descriptor (TWE/RWE). |
| 208 | INVLD\_KEY | One of the keys chosen for the operation is not valid or is not owned by the PCI function that owns the QP that initiated the operation. (KME) |
| 240 | DATA\_INTEGRITY\_RQE | Parity check failed on the RQE. (TWE) |
| 241 | DATA\_INTEGRITY\_AUX\_CMD | CRC check on AUX Command failed. (TXC) |
| 244 | DATA\_INTEGRITY\_RAW\_AUX | CRC check on input AUX\_FRAME\_DATA failed. (TXC) |

### SCH TLV

The Scheduler Update TLV indicates what the compression or decompression ratio is for some amount of data that has been consumed. The engine will generate these periodically for the operation when it finishes with each coding block. Please note that the SCH TLV is transferred over a separate 8-bit AXI interface that is described in 3.2.1.4. A description of how this TLV is time-multiplexed over the 8-bit interface is also included in 3.2.1.4.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ***Offset*** | ***31*** |  | | | | | ***0*** |
| 0x0 | unused(4) | | [eng](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_sched.htm#sched_tlv/NONE/NONE/eng) | [seq](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_sched.htm#sched_tlv/NONE/NONE/seq) | [length](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_sched.htm#sched_tlv/NONE/NONE/length) | | [type](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_sched.htm#sched_tlv/NONE/NONE/type) |
| 0x4 | [bip2](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_sched.htm#sched_tlv/NONE/NONE/bip2) | [command\_complete](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_sched.htm#sched_tlv/NONE/NONE/command_complete) | unused(2) | [scheduler\_handle](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_sched.htm#sched_tlv/NONE/NONE/scheduler_handle) | | [frame](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_sched.htm#sched_tlv/NONE/NONE/frame) | |
| 0x8 | unused(8) | | | [bytes\_out](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_sched.htm#sched_tlv/NONE/NONE/bytes_out) | | | |
| 0xc | unused(8) | | | [bytes\_in](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_sched.htm#sched_tlv/NONE/NONE/bytes_in) | | | |
| 0x10 | unused(8) | | | [basis](http://projects.irv.broadcom.com/HSC-NX1/creole_arch_docs_dev_5p7/arch/html/Top_chip_core_CCEIP0_Interfaces_sched.htm#sched_tlv/NONE/NONE/basis) | | | |

**Bits: 130 / Bytes: 17 / Unused Bits: 30**

#### type (Offset:0x0[7:0], Size: 8)

TLV Type - The value in this field defines the field format to apply to the entire TLV.

|  |  |  |
| --- | --- | --- |
| ***Value*** | ***Enum*** | ***Enumeration Description*** |
| 26 | sched | Scheduler Update TLV |

#### length (Offset:0x0[15:8], Size: 8)

Length of the TLV in units of 4 bytes. This length is computed over the entirety of the TLV Type, Length and data fields.

The length may indicate that only part of the data in the last beat is valid. The next TLV will start at bit zero of the interface. Unused area in the last beat will be sent as zero.

#### seq (Offset:0x0[23:16], Size: 8)

Per-Engine sequence number for the operation. This value will be incremented by one for each operation on an engine. All TLV for the same operation will have the same sequence number.

#### eng (Offset:0x0[27:24], Size: 4)

Engine ID for the operation.

#### frame (Offset:0x4[10:0], Size: 11)

This field will be zero for Scheduler Update TLV.

#### scheduler\_handle (Offset:0x4[26:11], Size: 16)

|  |  |  |
| --- | --- | --- |
| ***Offset*** | ***Field*** | ***Field Description*** |
| 15:13 | qg\_cre | This field carries the queue group compression ratio estimate.  This field is used to scale the initial outbound bytes charge in TWE, and the final outbound bytes refund in RXC.  For compression, the scale factor represents a negative exponent. I.e., the scale factor will be 2^(-qg\_cre).  For decompression, the scale factor represents a positive exponent. I.e., the scale factor will be 2^qg\_cre.   |  |  |  | | --- | --- | --- | | ***Value*** | ***Enum*** | ***Enumeration Description*** | | 0 | sf1 | Compression scale factor of 1.  Decompression scale factor of 1. | | 1 | sf2 | Compression scale factor of 1/2.  Decompression scale factor of 2. | | 2 | sf4 | Compression scale factor of 1/4.  Decompression scale factor of 4. | | 3 | sf8 | Compression scale factor of 1/8.  Decompression scale factor of 8. | | 4 | sf16 | Compression scale factor of 1/16.  Decompression scale factor of 16. | | 5 | unsupported32 | This scale factor is unsupported and must not be used. | | 6 | unsupported64 | This scale factor is unsupported and must not be used. | | 7 | unsupported128 | This scale factor is unsupported and must not be used. | |
| 12 | sg\_ob\_charged | If this bit is set, then outbound scheduling group meters are to be charged.  If this bit is clear, then outbound scheduling group meters are not to be charged. |
| 11 | sg\_ib\_charged | If this bit is set, then inbound scheduling group meters are to be charged.  If this bit is clear, then inbound scheduling group meters are not to be charged. |
| 10 | qg\_ob\_charged | If this bit is set, then outbound queue group meters are to be charged.  If this bit is clear, then outbound queue group meters are not to be charged. |
| 9 | qg\_ib\_charged | If this bit is set, then inbound queue group meters are to be charged.  If this bit is clear, then inbound queue group meters are not to be charged. |
| 8:0 | queue\_group | This value represents the tqme queue group. |

#### command\_complete (Offset:0x4[29], Size: 1)

This bit will be '1' for the last scheduler TLV of each command. This is used by RXC to know when to flush out the fractional bytes due to scaling.

#### bip2 (Offset:0x4[31:30], Size: 2)

2bit parity value over the first 64b of the TLV. bip2[1] contains xor computed over the odd bits in the first 64b word of TLV excluding bip2[1]. bip2[0] contains xor computed over the even bits in the first 64b word of TLV excluding bip2[0].

#### bytes\_out (Offset:0x8[23:0], Size: 24)

Data output bytes generated by the engine. The difference between this basis and this value will be adjusted into the output cost for the user that generated this command.

#### bytes\_in (Offset:0xc[23:0], Size: 24)

Data input bytes consumed by the engine. The difference between the basis and this value will be adjusted into the input cost for the user that generated this command.

#### basis (Offset:0x10[23:0], Size: 24)

Data input bytes passed into the engine for this portion of the ratio calculation. This is the actual data input size which must be a portion of the input data size provided in the RQE. This value should not inclucde any extra input cost such as the predefined prefix.

## Project Zipline Command Types and Allowable Header Stack ups

Within these canonical TLVs, there are a finite number of stack-ups that are allowed, the following figures represent the legal stackups that can be presented to the CCEIP engine interfaces.

* Note: TLVs that the CCEIP/CDDIP do not parse will be passed through unmodified, in the same relative position that they came into the engine

Figure : Allowable Header Stack ups for Simple Command



Figure : Allowable Header Stackups for Compound Commands



### Top Level Interfaces

The top level interfaces are specified as AMBA compliant-

* The control processor interface is APB3.0
* All other datapaths are AXI-Streaming

#### Control Plane Interface

* An APB Interface will be provided to provide for Configuration & Status Register (CSR) access as well as various debug facilities
* This APB Interface will contain a bus-timeout feature, nominally provisioned for 1ms.
* The APB bus is synchronous to the rest of the CCEIP/CDDIP logic (meaning there is a single clock).

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **I/O** | **Width** | **Description** |
| apb\_psel | I | 1 | Peripheral Select |
| apb\_penable | I | 1 | Peripheral Enable |
| apb\_addr | I | 120 | Address Bus |
| apb\_pwdata | I | 32 | Write Data Bus |
| apb\_write | I | 1 | Write Enable (Write=1/Read=0) |
| apb\_prdata | O | 32 | Read Data Bus |
| apb\_pready | O | 1 | Read Data Valid/ Write has been accepted |
| apb\_pslverr | O | 1 | Read Error |

#### Inbound Data Path

* This is a 64 bit AXI Streaming Datapath

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **I/O** | **Width** | **Description** |
| ib\_tid | I | 1 | Block Identifier |
| ib\_tvalid | I | 1 | Data Beat is valid |
| ib\_tstrb | I | 8 | Data Byte is valid on the bus  0: Bits 7:0  1: Bits 15:8  …  7: Bits 63:56 |
| ib\_tlast | I | 1 | Set on last beat of the Command’s Completion Message |
| ib\_tdata | I | 64 | Datapath – see header stackup |
| ib\_tuser | I | 8 | 8 bits defined  0: Start of TLV  1: End of TLV  2-7 Reserved |
| ib\_tready | O | 1 | Validates the data acceptance by CCEIP/CCDIP.  0 : Not Accepted  1: Accepted |

#### Outbound Data Path

* This is a 64 bit AXI Streaming Datapath

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **I/O** | **Width** | **Description** |
| ob\_tid | O | 1 | Block Identifier |
| ob\_tvalid | O | 1 | Data Beat is valid |
| ob\_tlast | O | 1 | Set on Completion Message last beat |
| ob\_tstrb | O | 8 | Data Byte is valid on the bus  0: Bits 7:0  1: Bits 15:8  …  7: Bits 63:56 |
| ob\_tdata | O | 64 | Datapath – see header stackup |
| ob\_tuser | O | 8 | 8 bits defined; undefined bits are reserved  OB\_TUSER[0] : Start of TLV  OB\_TUSER[1] : End of TLV  Others: Reserved |
| ob\_tready | I | 1 | Validates the data acceptance by CCEIP/CCDIP.  0 : Not Accepted  1: Accepted |

#### Scheduler Update Interface

* This is a 64 bit AXI Streaming Datapath

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **I/O** | **Width** | **Description** |
| sch\_update\_tvalid | O | 1 | Data Beat is valid |
| sch\_update\_tlast | O | 1 | Asserted on last beat of each Scheduler message. |
| sch\_update\_tuser | O | 2 | 0: Start of update  1: End of update  2-3 Reserved |
| sch\_update\_tdata | O | 8 | Data |
| sch\_update\_tready | I | 1 | Validates the data acceptance by CCEIP/CCDIP.  0: Not Accepted  1: Accepted |

The table below describes how the Scheduler Update TLV is time multiplexed over the 8-bit AXI interface.

|  |  |
| --- | --- |
| **8-bit Beat** | **Contents** |
| 0 | type |
| 1 | length |
| 2 | seq |
| 3 | {unused, eng} |
| 4 | frame[7:0] (all 0’s) |
| 5 | {scheduler\_handle[4:0,] frame[10:8]} (frame all 0’s) |
| 6 | scheduler\_handle[12:5] |
| 7 | {bip2, command\_complete,unused,scheduler\_handle[15:13]} |
| 8 | bytes\_out[7:0] |
| 9 | bytes\_out[15:8] |
| 10 | bytes\_out[23:16] |
| 11 | unused |
| 12 | bytes\_in[7:0] |
| 13 | bytes\_in[15:8] |
| 14 | bytes\_in[23:16] |
| 15 | unused |
| 16 | basis[7:0] |
| 17 | basis[15:8] |
| 18 | basis[23:16] |
| 19 | unused |

#### Key Interface

* This is a 64 bit AXI Streaming Datapath

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **I/O** | **Width** | **Description** |
| kme\_tid | I | 1 | Block Identifier |
| kme\_tvalid | I | 1 | Data Beat is valid |
| kme\_tlast | I | 1 | Last Beat of the TLV |
| kme\_tdata | I | 64 | Key TLV |
| kme\_tstrb | I | 8 | Data Byte is valid on the bus  0: Bits 7:0  1: Bits 15:8  …  7: Bits 63:56 |
| kme\_tuser | I | 8 | 8 bits defined; undefined bits are reserved  OB\_TUSER[0] : Start of TLV  OB\_TUSER[1] : End of TLV  Others: Reserved |
| kme\_tready | O | 1 | Validates the data acceptance by CCEIP/CCDIP.  0 : Not Accepted  1: Accepted |

* The KEY TLV is the only TLV type on the KEY Interface

# Detailed Block Flow Walkthrough

The following is a detailed walkthrough of the manipulation of the data and TLV headers as they flow through each block.

For all TLV processing by the sub-blocks the error procedure is the same:

1. The FOOTER TLV has a field to indicate that an error occurred (FIRST ERROR). When a block encounters an error it first checks to see if this field is non-zero.
   * If the First Error field is 0, the block with the error will modify this field with the appropriate error code.
   * If the First Error field is non-zero, it will not be overwrite this field. The intent is to only report the first error found in the engine pipeline.
2. All the blocks below can report errors via the CQE message.

### CCEIP Command Processing

#### Inbound Streaming Fifo

The ISF manipulates the datapath in the following ways:

1. To support the debug functionality in the CCEIP the ISF will look at the various APB programmable registers to implement the breakpoint and single-step functionality
   * Breakpoints can be created by programming the ISF to freeze on patterns within the incoming TLVs (see Shared Support block for more details regarding capabilities).
   * Once a breakpoint has been reached, the datapath can be single stepped.
2. The ISF does the job of breaking the opaque data TLV into 2 TLVs. It will look at the CMD.Compression.Prefix field, and if the user prefix is present will divide the DATA TLV into 2 TLVs:

* User Prefix TLV (It will generate the TLV framing structure, length, etc) along w/ the data
* Data TLV Length adjustment (to subtract out the User Prefix size).

The ISF will can be presented with several different frame stackups – it manipulates them as follows:

Figure : ISF Simple Command TLV Manipulation

1. RQE, CMD and the incoming FRMD-TLV will be copied as is to the ISF Outbound datapath.
2. The ISF will insert the FOOTER TLV and STATS TLV into the stackup prior to the incoming CQE TLV.
3. The ISF will copy the CMD.Compression.XP10PrefixSelect to the FRMD for all FRMDs (simple and compound) for the prefix engine.
4. The ISF will copy fields from the FRMD and CMD TLV to pre-populate the FOOTER TLV with all of the control information that it needs.

The Footer will be pre-populated as follows:

|  |  |  |
| --- | --- | --- |
| **Field** | | **Source** |
| Type | | Hardcoded FOOTER TLV Type |
| Length | | Hardcoded FOOTER TLV Length |
| Engine ID | | From FRMD TLV Word 0 |
| Sequence Num | | From FRMD TLV Word 0 |
| Frame Num | | From FRMD TLV Word 0 |
| Reserved | | From FRMD TLV Word 0 |
| Parity | | Recomputed over new Word 0 |
| GUID | | 0 |
| IV | | 0 |
| Status | Coding | Copied from Command TLV field |
| RawDataCkMACType | Copied from Command TLV field |
| EncCmpCkMACType | Copied from Command TLV field |
| Reserved | 0 |
| GenFRMDOutType | | Copied from Command TLV field |
| Compressed Length | | 0 |
| Uncompressed Length | | Copied from Data TLV Length |
| RawDataDigest | | 0 |
| RawDataCkSum | | 0 |
| EncCmpDataDigest | | 0 |
| EncCmpDataCkSum | | 0 |
| Error Code | | 0 |
| Errored Frame Number | | 0 |
| Reserved | | 0 |

##### Differences between Simple and Compound Command Processing:

* The ISF needs to insert a footer after each DATA TLV, regardless if it is compound command or simple command.
* The ISF needs to generate latency information for the first frame in a compound command only, and the for the only frame in a simple command.

#### RawData Checksum Generator/Checker

This block will build a CRC-64 over the Frame Data specified in the DATA TLV. This is over Raw / uncompreseed/un-encrpyted data. This CRC does not cover the User Prefix (as it has been segmented off of the Data into its own TLV).

* The Polynomial is the XP10 CRC64 polynomial:
  + x^64 + x^63 + x^61 + x^59 + x^58 + x^56 + x^55 + x^52 + x^49 + x^48 + x^47 + x^46 + x^44 + x^41 + x^37 + x^36 + x^34 + x^32 + x^31 + x^28 + x^26 + x^23 + x^22 + x^19 + x^16 + x^13 + x^12 + x^10 + x^9 + x^6 + x^4 + x^3 + 1
* The handling is as follows:
  + Footer.RawDataCkSum is always updated with the newly generated CRC
  + If the Footer.RawDataCkSum is !=0 and doesn’t match the newly generated CRC the logic will generate an error and apply to the FOOTER Error Field.

Figure : RawData Checksum Generator/Checker TLV Manipulation



* The CRC generator computes the CRC and populates in the FOOTER.RawDataCkSum field.
* If the Incoming FRMD has a valid CRC, this block will compare what it generates vs the incoming CRC and if necessary generate an error in the FOOTER.

##### Differences between Simple and Compound Command Processing:

* None

#### Prefix Engine

The prefix engine will act as follows:

|  |  |
| --- | --- |
| **XP10PrefixSel** | **Action** |
| 0 | Run Prefix Engine on first 4KB of frame, place result in **Prefix Word 1 XP10PrefixSel** |
| Non Zero | Do not run prefix engine, software has specified a predetermined prefix. |

The Prefix Engine, if it detects and error will place an error code in the Completion message and abort processing.

Figure : Prefix Engine TLV Manipulation (Simple)



* The Prefix Engine updates the selected Prefix in the PREFIX TLV and emits a Prefix TLV stub that is filled in by the Prefix Attach block.

Table 3: Prefix Engine TLV Manipulation (Compound Command)



##### Differences between Simple and Compound Command Processing:

* This block will always write the selected PREFIX into the FRMD field.

#### Prefix Attach Engine

The Prefix Attach Block will examine the AUX\_CMD.Compress.XP10PrefixSel field, and AUX\_CMD.Compress.XP10PrefixMode fields to see which Prefix TLVs it will insert into the header stream:

Table 4: Prefix Attach Engine Command Processing

|  |  |  |
| --- | --- | --- |
| **AUX\_CMD.**  **XP10PrefixMode** | **AUX\_CMD.**  **XP10PrefixSel** |  |
| 0 | N/A | Do not insert any TLVs |
| 2 | 0 | Do not insert any TLVs |
| 3 | 0 | Do not insert any TLVs |
| 1 | N/A | User Defined Prefix is Present, do not insert any TLVs. |
| 2 | 1-63 | Insert Prefix TLV |
| 3 | 1-63 | Insert Predetermined Huffman and Prefix TLVs |

Note:

The Prefix Attach Engine, when in Predetermine Huffman + Prefix Mode will ALWAYS insert the Predetermined Huffman TLV before the Prefix TLV.

Figure : Prefix Attach TLV Manipulation



* The Prefix Attach Engine will insert the Predetermined Huffman and Predetermined Prefix TLVs.

##### Differences between Simple and Compound Command Processing:

* There are no difference between Simple and Compound Command processing. This block will store the Prefix Mode from the CMD header, and then look at the selected prefix in the FRMD header to make a decision.

#### Compression Engine (LZ77 Compressor)

The Compression Engine is fragmented into 2 pieces, the LZ77 Compress and Huffman Encoder blocks. The LZ77 Compressor does the following:

Uses the following fields:

|  |  |
| --- | --- |
| **CMD.Compression.CompMode** | 0: Bypass the compression engines  1: Engage the compression engines. |
| **CMD. Compression.LZ77WinSize** | Set length of history buffer window |
| **CMD. Compression.LZ77DelayMatchWin** | None, 1 or 2 Characters. |
| **CMD. Compression.LZ77MinMatchLen** | 0: Minimum 3 Byte Matches  1: Minimum 4 Byte Matches |
| **CMD. Compression.LZ77MaxMatchLen** | 0: Window Size  1: LZ77MinMatchLen+14  2: 258B  3: 64B |
| **CMD.XP10PrefixMode**  **CMD.Compression.XP10PrefixSel** | Used to control emission of data that is prefix data. |

The Compression Engine will build properly formatted LZ77 Symbols (see the Project Zipline\_comp\_lz77\_uarch document for implementation details). There are, however, additional frame handling details that are outlined below:

1. The Compression engine will generate a CRC of the incoming DATA TLV as it leaves the history buffer and compare against the FRMD-TLVs RawCkSum field using the method requested in the AUX\_FRMD.RawCKSumMode.
2. The Compression engine will check the CRC which is on the Prefix Data (User and Predetermined) and generate an error in the Completion Message if the CRC mismatches.
   * Note: User Prefix and Predetermined Prefix data both have a CRC-32 protecting them.
3. The Compression engine will remove the DATA TLV from the data stream and generate a Huffman-Data TLV which is a representation of the compressed data using Literals and LZ77 Symbols (Ptr/Offset) as well as MSFT MTFs (Reference/Length) tuples. This TLV can transmit up to 4 symbols per clock (but only 1 symbol can be a PTR or MTF).
   * Note: This implies that a frame full of 3-byte matches will fall below 25Gbps line rate. This is considered a pathological frame for which line rate is not expected. Uncompressible frames (all literals) will be transmitted at line rate.
4. The LZ77 Compressor uses the standard TLV Parser, so it has access to all the standard error generation capabilities.

Figure : LZ77 Compressor TLV Manipulation 

* The LZ77 Compressor will transform the DATA TLV into the LZ77- Symbol TLV.

#### Huffman Encoder

The Huffman Encoder block will receive the standard list of TLVs, except that the DATA TLV has been replaced with the Huffman-Debug TLV, used between the LZ77 and Huffman Encoder engines.

The details of the Huffman Encoder implementation are available in the associated Micro-architecture spec (Project Zipline\_Huffman\_Encoder\_Top). However, there are several processing details that should be noted here:

1. Coding Blocks are delineated by counting Symbols (Literals/MTF/PTRs); when either the Frame ends or 8K Symbols have been detected the coding block ends. This affects when the Retrospective Huffman Tables are built and the XP9/XP10/Deflate envelope generation.
   * For MSFT: This is a slight departure from the MAS uarch, where coding blocks are delineated by counting short/long/literal symbols.
2. The Huffman Encoder uses the CMD.Compression header to decide the type of coding used (XP9/XP10/GZIP-Deflate/ZLIB-Deflate/CHU). It also can emit RAW frames without any envelope.
   * The Huffman Encoder will output the User and Predetermined Prefixes
   * The Huffman Encoder will output this data as a DATA TLV
   * The Huffman Encoder will update the FOOTER TLV Status field’s UNCOMPRESSIBLE bit.
3. When the Huffman Encoder builds the XP10 or XP10CFH4K/8K it shall populate the Prefix Number field with the Prefix TLVs Selected Prefix Number which is in Word 1 of the Prefix.
   * Note: both the Predetermined Huffman tree and Prefix TLV contain the same value

Figure : Huffman Encoder TLV Manipulation 

* The Huffman Encoder compresses the LZ77 Symbol TLV and outputs a DATA TLV of compressed data coding blocks. The FOOTER TLV’s Status.Coding field is updated depending on the mode: Raw, Parsable or CHU4K, CHU8K.
* The Huffman Encoder needs to consume the CMD, PreHuff, Prefix, LZ77Sym and footer TLVs.

#### Crypto Engine

The Cryptography and Authentication engine will encrypt and authenticate the compressed frame data. The Crypto Engine receives information from 2 places:

* Main Datapath (fed from the Compression Engine’s Huffman Encoder)
  + CMD TLV – tells the engine the mode it is running in.
  + FRMD TLV – gives encryption engine the IV (if needed).
  + Data TLV – the actual data to encrypt
* Key AXI Streaming Interface (fed from the KME).
  + KEY TLV – provides keys and GUID

The Crypto engine works as follows:

1. A CMD TLV and KEY TLV are paired up with matching sequence IDs via the pair of AXI Streaming interfaces. The Crypto engine will wait until a FIFO fed from each interface has the same Sequence ID and passes the engine ID check. If they do not match the Crypto engine will signal an error on this command.
   1. For each Command (RQE) there will be a KEY TLV, regardless if the compression and authentication engines are used. This is necessary to ensure robust sequence ID handling by forcing a monotonic incrementing scheme for commands in ALL use cases.
   2. If these interface backpressures long enough, the KME will ultimately backpressure as well and all traffic to ALL CCEIPs/CDDIPs will be halted (aka there is ultimately head-of-the-line blocking back to the TXC via the KME).
   3. All Crypto/Authentication engines are fed with the same KEY buses.
2. The KEYs will be expanded as per the algorithm selection.
3. The IV needs to be generated per the selected mode in the CMD TLV. The options are:
   1. **Fixed** IV (used for decryption), where the IV is contained in the FRMD TLV that precedes the Data TLV. This can be used for Simple and Compound Commands.
   2. **Random** Generated (used for encryption), where the IV is generated per frame. This can be used for Simple and Compound Commands.
   3. **Incrementing**, where the initial IV is provided from the FRMD TLV that precedes the first data frame in a compound command and is incremented for each subsequent frame. This is used for encryption and decryption.
4. The Crypto block needs to check that the engine ID fields and sequence numbers in the TLVs all match to validate that the KEY being used on the frame is the correct one. Encryption takes place first using an AES engine capable of sustaining 4 Rounds per clock using AES-GCM/XTS/XEX algorithms.
5. Authentication takes place after Encryption using a SHA2 engine which can run at 25Gbps for SHA2 and SHA2-HMAC. The Authentication tag is placed in the FOOTER TLV.
   * + Performance Note: The SHA2-HMAC algorithm has 4 additional 64 byte blocks that need to be authenticated. As such, to meet the line rate operation, the design will actually instantiate a pair of SHA2 engines which work in series as a super-pipeline (the first SHA engine will do the first 32 blocks), then pass the residual tag to the second engine to complete the final 4 rounds.

Figure : Encryption and Authentication TLV Manipulation (Simple Commands)



1. The Crypto Engine is used to encrypt the data and create the authentication tag that is present in the FRMD TLV.
2. The IV mode is defined in the CMD TLV and an initial IV or an explicit IV will be in the FRMD data.
3. The GUID is copied from the KEY TLV to the Footer TLV TLV along w/ the IV.
4. Key material is presented on the KEY Interface, whereas the data and IV is present on the main datapath.



#### CRC Generator (Encrypted/Compressed Data)

The frame data post compression and encryption will have a CRC64 protecting it (CmpEncCkSum) generated and placed in the Footer.

The CRC64 Polynomial is the XP10 CRC64

* + x^64 + x^63 + x^61 + x^59 + x^58 + x^56 + x^55 + x^52 + x^49 + x^48 + x^47 + x^46 + x^44 + x^41 + x^37 + x^36 + x^34 + x^32 + x^31 + x^28 + x^26 + x^23 + x^22 + x^19 + x^16 + x^13 + x^12 + x^10 + x^9 + x^6 + x^4 + x^3 + 1

Figure : Encrypted/Compressed Checksum Generator 

* The Encrypted/Compressed CRC Generator builds a CRC-64 of the encrypted data and places it in the Footer TLV.

#### CRC Checker (Encrypted/Compressed Data)

The frame data and the CRC generated in Section 1.5.1.8 will be checked and an error code posted to the Completion Message. This is the CmpEncCkSum using the XP10 CRC64 polynomial:

* + x^64 + x^63 + x^61 + x^59 + x^58 + x^56 + x^55 + x^52 + x^49 + x^48 + x^47 + x^46 + x^44 + x^41 + x^37 + x^36 + x^34 + x^32 + x^31 + x^28 + x^26 + x^23 + x^22 + x^19 + x^16 + x^13 + x^12 + x^10 + x^9 + x^6 + x^4 + x^3 + 1

Figure : Encrypted/Compressed Checksum Checker 

* The Encrypted/Compressed CRC Checker validates the CRC-64 of the encrypted data that is in the FOOTER TLV.
* Any errors that are detected are added to the FOOTER TLV.

#### Decryption and Authentication Engine

This is a reverse process of the Encryption engine. Of note the following:

* Authentication will check the tag present in the Footer and signal an error when it is not valid in the Completion Message.

Figure : Decryption and Authentication Validation Engine TLV Manipulation



1. The Crypto Engine is used to decrypt the data and validate the authentication tag that is present in the FOOTER TLV.
2. Key material is presented on the KEY Interface, whereas the data and IV is present on the main data path.
3. IV processing is the same as for the encryption engine.

#### Decompression (Huffman Decoder)

The decompression engine takes in the TLV stackup and will perform Huffman and LZ77 decompression process. The Decompressor block terminates the Predetermined Huffman and Prefix TLVs.

* The Decompressor will check the CRCs present on the Predetermined Huffman and Prefix TLVs and place an error code in the Completion TLV and halt processing if there is an error.
* The Decompressor, when it detects the Prefix TLV moves that data directly to the LZ77 History Buffer.
* The Decompressor needs to look at the FRMD TLV and the beginning of the actual DATA TLV in order to make a determination about how to process the data along with the CMD TLV . **For decompression, the CMD.Compress fields will not be used**

Figure : Huffman Decoder TLV Manipulation 

1. The Huffman Decoder terminates the Predetermined Huffman TLV.
2. The DATA TLV is transformed into an LZSYM-TLV

Of note is the handling to decompress frames. The decompressor needs to parse the Data TLV and FRMD TLV to understand the formats rather than the AUX\_CMD.Compression fields to interpret the data.

The processing is:

* If FRMD.Status
  + 00: Raw
  + 01: Parsable (See table below)
  + 10: CHUXP10CFH 4K
  + 11: CHUXP10CFH 8K

|  |  |  |  |
| --- | --- | --- | --- |
| **Mode** | **FRMD.Type** | **FRMD.Status.**  **Uncompressible** | **Data[31:0]** |
| Raw Data | All | 1 | N/A |
| CFH 4K | ChuInplace Type | 0 | N/A |
| CFH 8K | CHUInplace Type | 0 | N/A |
| XP9 | CHUAppend  VM | 0 | 0x4e86d72a  *Magic Number* |
| XP10 | CHUAppend  VM | 0 | 0xC039E510  *Magic Number* |
| ZLIB | CHUAppend  VM | 0 | 0x1D08 FDICT=0, Compression = 0  0x5B08 FDICT=0, Compression = 1  0x9B08 FDICT=0, Compression = 2  0xDB08 FDICT=0, Compression = 3  *CMF Only:*  *0x\*\*\*8*  *CMF with FCHECK (FDICT=0 and all 4 compression levels):* |
| GZIP | CHUAppend  VM | 0 | 0x8B1F  *GZIP ID1 and ID2 fields.* |

Window Size will be extracted as:

|  |  |
| --- | --- |
| Frame Type | Window Size |
| Raw | N/A |
| CHU4K | 4K |
| CHU8K | 8K |
| XP10 | XP10.FrameHeader.Flags.Window\_Size\_Sel |
| XP9 | XP9.Header.Win\_Size |
| ZLIB | ZLibHeader.CINFO |
| GZIP | 32K |

#### Decompression (LZ77)

The LZ77 Decompressor uses the Prefix Data (either the 1KB Predetermined Prefix or the 1-64KB User Prefix Data to prime the LZ77 Window, and then performs the symbol expansion of the LZ77 Symbols.

* The Huffman Decoder block will perform the MTF to PTR mapping
* The LZ77 Symbol Expander will have a peak performance of 50Gbps due to the size of the outgoing datapath (64 bits @ 800MHz).
* The XP9/XP10/Deflate Adler will be checked post decompression in this engine.

Figure : LZ77 TLV Manipulation



1. The LZ77 Decompressor will terminate the Prefix TLV.
2. The LZ77 Decompressor will transform the LZSYM TLV back into Data.
3. Appropriate CRCs will be checked and any errors propagated to the FOOTER and CQE TLVs.
4. The Uncompressed data size will be updated in the STATS TLV

#### CRC Checker (Raw Data)

This block will check 2 CRCs that cover the Raw Data. They operate as follows:

1. RawDataCkSum

This CRC was either gethe raw CRC over the Frame Data specified in the DATA TLV. This is over Raw / uncompreseed/un-encrpyted data. This CRC does not cover the User Prefix (as it has been segment off of the Data into its own TLV).

* + x^64 + x^63 + x^61 + x^59 + x^58 + x^56 + x^55 + x^52 + x^49 + x^48 + x^47 + x^46 + x^44 + x^41 + x^37 + x^36 + x^34 + x^32 + x^31 + x^28 + x^26 + x^23 + x^22 + x^19 + x^16 + x^13 + x^12 + x^10 + x^9 + x^6 + x^4 + x^3 + 1

Figure : Raw CRC Checker TLV Manipulation



1. The Raw CRC checker will compute the CRC and compare against the CRC that is in the Footer
2. Errors are reported in the CQE

#### Integrity/Authentication (Crypto) Engine

The final datapath check that will be created is an integrity/authentication check on the Raw Data. This is done at the end of the pipeline by a version of the Crypto engine that does not have the AES core. This authentication check will be either SHA2 or SHA2-HMAC. The authentication tag is placed in the Footer TLV.RawMAC field

Figure : Raw Integrity and Authentication Engine



* The Authentication engine consumes a KEY TLV that is presented on the KEY interface and uses the main Datapath to build an authentication tag which is placed in the FRMD-OUT-TLV.

#### Outbound Streaming Interface

The outbound streaming interface taps into the datapath as it transits from Compressed CRC Geneator to Compressed CRC Checker. It copies the DATA TLV to a fifo that drives the Outbound streaming interface.

#### Completion Streaming Interface

The completion streaming interface builds the final outgoing FRMD, STATS and CQE TLVs. These data structures are carried through the CCEIP and CDDIP engines and need to be finalized. They are scheduled on the outgoing outbound streaming interface.

Figure : Completion Streaming Interface



1. Builds the selected FRMD by using fields from FRMD-OUT and the NMD-TLV.
2. Finalizes the STATs and CQE TLVs
3. Terminates the other TLVs that have been carried through the pipeline.

The Completion Interface works as follows:

* + Stats TLV is transmitted with the final latency of the first frame
  + Completion TLV is updated by taking the Error fields from the Footer and moving them to the Completion TLV.
  + The FRMD is built by interrogating the FOOTER.FRMD\_OUT\_TYPE field and building a TLV of appropriate length and pulling fields from the Footer and putting into fixed locations in the FRMD.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | | **Size** | **Word** | **Notes** |
| Type | | 8 | 0 | See Project Zipline TLV Type Reference Table  Type = 6 |
| Length | | 8 | 0 | Length = 28 (Length of mini-FOOTER from ISF) |
| Sequence Number | | 8 | 0 | Increments per RQE/Command |
| Engine ID | | 4 | 0 | 0-3 CCEIP, 4-7 CDDIP |
| Reserved | | 4 | 0 | Reserved |
| Frame Number | | 11 | 0 | 0 – Simple Command  0-2047 for Compound Commands |
| Reserved | | 1 | 0 | Reserved |
| Status | Coding | 2 | 0 | 0: Raw  1: Parsable  2: XP10CFH 4K  3: XP10CFH 8K |
| EncCmpDataMacSize | 2 | 0 | 0: 64 bits valid (upper 64 bits of the field)  1: 128 bits valid (upper 128 bits of the field)  2: 256 bits valid (full size of the field)  3: Reserved |
|  | RawDataMACSize | 2 | 0 | 0: 64 bits valid (upper 64 bits of the field1: 128 bits valid (upper 128 bits of the field)  2: 256 bits vaelid (full size of the field)  3: Reserved |
|  | Reserved | 2 | 0 | Reserved |
| GenFRMDOutType | | 8 | 0 |  |
| Reserved | | 2 |  |  |
| Parity | | 2 | 0 | BIP-2 (Even) |
| GUID | | 256 | 1-4 |  |
| IV | | 128 | 5-6 | Big Endian format with bits [127:64] in Word5 |
| RawDataMAC | | 256 | 7-10 | SHA2, SHA2-HMAC (Big Endian format with bits [255:192] in Word7) |
| RawDataCkSum | | 64 | 11 | CRC-64 |
| RawDataCkSumProtocol | | 64 | 12 | CRC added to the compressed data in the Huffman Encoder for XP10, GZIP, and ZLIB,. CRC can be one of has 4 different types per the protocol specified in CMD. comp\_ctl.algorithm and the CRC specified in CMD. comp\_ctl.xp10\_crc\_mode and described in Table 18. |
| EncCmpDataMac | | 256 | 13-16 | SHA2, SHA2-HMAC, AES-GMAC (Big Endian format with bits [255:192] in Word13) |
| EncCmpDataCkSum | | 64 | 17 | CRC-64 |
| Bytes Out | | 24 | 18 | Bytes Out |
| Bytes In | | 24 | 18 | Bytes In (xfer’d over PCIe) (ISF updates) |
| NVMERawCkSum | | 16 | 18 | CRC-16T |
| Errored Frame Number | | 11 | 19 | 0 for simple commands; otherwise has the first frame in the compound command that generated the error. |
| Reserved | | 1 | 19 | Reserved |
| Error Code | | 8 | 19 | Engine error code (see [2] for a complete list) |
| Compressed Length | | 24 | 19 | Compressed Size (Huffman Encoder builds) |
| Reserved | | 20 | 19 | Reserved |

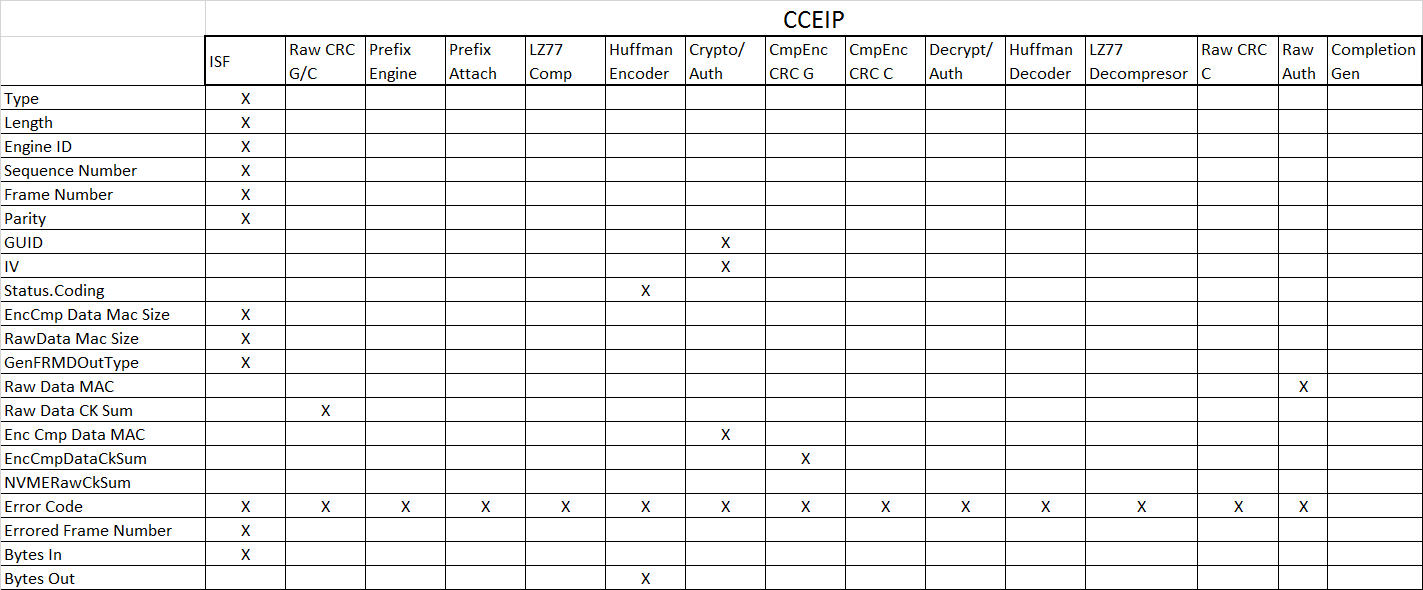
Table 5: Footer Fields at Completion Streaming Interface

|  |  |  |
| --- | --- | --- |
| ***Error Code*** | ***Name*** | ***Description*** |
| 0 | NO\_ENGINE\_ERRORS | No Errors detected during Compression/Encryption Command |
| 1 | HD\_MEM\_ECC | Decoder ECC Failure |
| 2 | HD\_FHP\_PFX\_CRC | Decoder Predetermined Prefix CRC Mismatch |
| 3 | HD\_FHP\_PFX\_DATA\_ABSENT | Decoder Expected Prefix and didn't receive one |
| 4 | HD\_FHP\_PHD\_CRC | Decoder Predetermine Huffman Tables CRC Mismatch |
| 5 | HD\_FHP\_BAD\_FORMAT | Decoder Bad in Format Bad |
| 6 | HD\_BHP\_INVALID\_WSIZE | Decoder Received invalid window size in the Frame header for XP10 |
| 7 | HD\_BHP\_BLK\_CRC | Decoder XPx BLK CRC failure |
| 8 | HD\_BHP\_HDR\_INVALID | Decoder Received Invalid XPx Header (bits not 0) |
| 9 | HD\_BHP\_XP9\_HDR\_SEQ | Decoder XP9 Hdr Sequencer number not 0 |
| 10 | HD\_BHP\_XP10\_XTRA\_FLAG\_PRSNT | Decoder Received XP10 with XTRA FLAG Set (not supported) |
| 11 | HD\_BHP\_ZLIB\_FDICT\_PRSNT | Decoder ZLIB FDICT indicates Present (not supported) |
| 12 | HD\_BHP\_GZ\_CM\_NOT\_DEFLATE | Decoder GZIP header indicates deflate wasn't used as compression algorithm |
| 13 | HD\_BHP\_ZLIB\_CINFO\_RANGE | Decoder CINFO Out of Range |
| 14 | HD\_BHP\_ZLIB\_FCHECK | Decoder FCHECK check failed for ZLIB frame. |
| 15 | HD\_BHP\_DFLATE\_LEN\_CHECK | Decode LEN and NLEN check failed for RAW deflate block. |
| 16 | HD\_LFA\_REWIND\_FAIL | Deocoder Internal Error during Rewind - invalide Huffman Symbols |
| 17 | HD\_LFA\_PREMATURE\_EOF | Decoder Premature in Eof Premature |
| 18 | HD\_LFA\_LATE\_EOF | Decoder Late in Eof Late |
| 19 | HD\_LFA\_MISSING\_EOF | Decoder EOF not received at the end of last block |
| 20 | HD\_HTF\_XP9\_RESERVED\_SYMBOL\_TABLE\_ENCODING | Decoder received invalid XP9 symbol table encoding |
| 21 | HD\_HTF\_XP10\_RESERVED\_SYMBOL\_TABLE\_ENCODING | Decoder received invalid XP10 symbol table encoding |
| 22 | HD\_HTF\_XP10\_PREDEF\_SYMBOL\_TABLE\_ENCODING | Decoder received invalid XP10 predetermined symbol table encoding |
| 23 | HD\_HTF\_XP9\_ILLEGAL\_NONZERO\_BL | Decoder received an illegal non-zero bit length |
| 24 | HD\_HTF\_RLE\_OVERRUN | Decoder encounter a run length encoding overrun condition |
| 25 | HD\_HTF\_BAD\_HUFFMAN\_CODE | Decoder Bad in Code Huffman |
| 26 | HD\_HTF\_ILLEGAL\_SMALL\_HUFTREE | The block's unpacked short symbol table contains errors |
| 27 | HD\_HTF\_ILLEGAL\_HUFTREE | The block's unpacked long symbol table contains errors |
| 28 | HD\_HTF\_HDR\_UNDERRUN | Huffman header length is less than specified |
| 29 | HD\_BHP\_STBL\_SIZE\_ERR | Decoder symbol table size in the header does not match actual size. |
| 32 | HD\_SDD\_INVALID\_SYMBOL | Decoder Invalid in Symbol Invalid |
| 33 | HD\_SDD\_END\_MISMATCH | Decoder End in Mismatch End |
| 34 | HD\_SDD\_MISSING\_EOB\_SYM | Decoder Missing in Sym Eob |
| 35 | HD\_MTF\_XP9\_MTF3\_AFTER\_BACKREF | Decoder Xp9 Mtf3 in Backref After |
| 36 | HD\_MTF\_XP10\_MISSING\_MTF | Decoder Xp10 in Mtf Missing |
| 37 | HD\_BHP\_ILLEGAL\_MTF\_SZ | Decoder received MTF offsets greater than 16 bits |
| 38 | HD\_LZ\_HBIF\_SOFT\_OFLOW | Decompressor Fifo Overflow |
| 39 | HD\_BE\_FRM\_CRC | Decompressor CRC failed on Decompression (ADLER/XP9) |
| 40 | HD\_BE\_OLIMIT | Decompressor expanded beyond 8MB |
| 41 | HD\_BE\_SZ\_MISMATCH | Decomprssor expanded size doesn't match header expected size |
| 50 | CRCG\_CRC\_ERROR | Raw CRC Checker Error |
| 51 | CRCC0\_CRC\_ERROR | The CCEIP engine detected a CRC error that was calculated over the encrypted and compressed data, as output by the compressor. |
| 52 | CRCC1\_CRC\_ERROR | The CCEIP engine detected a CRC error that was calculated over the raw data, as output by the validator. |
| 54 | CRCGC0\_CRC\_ERROR | The CCEIP engine detected a CRC error that was calculated over the raw data, as input to the compressor. |
| 56 | CRCDC0\_CRC\_ERROR | The CDDIP engine detected a CRC error that was calculated over the encrypted and compressed data, as input to the decompressor. |
| 64 | LZ77\_COMP\_PREFIX\_CRC\_ERROR | Prefix CRC Check failed during LZ77 Compressor loading. |
| 65 | LZ77\_COMP\_INVALID\_COMP\_ALG | AUX\_CMD Compression Algorithm undefined code point |
| 66 | LZ77\_COMP\_INVALID\_WIN\_SIZE | AUX\_CMD Compression Algorithm Window Size undefined code point |
| 67 | LZ77\_COMP\_INVALID\_MIN\_LEN | AUX\_CMD Compression Algorithm Minimum Match Size undefined code point |
| 68 | LZ77\_COMP\_INVALID\_NUM\_MTF | Compression Algorithm Number of MTF undefined code point |
| 69 | LZ77\_COMP\_INVALID\_MAX\_LEN | Compression Algorithm Maximum Match Length undefined code point |
| 70 | LZ77\_COMP\_INVALID\_DMW\_SIZE | Compression Algorithm Delayed Match Window undefined code point |
| 71 | LZ77\_COMP\_LZ\_ERROR | LZ77 match state machine error detected |
| 80 | HE\_MEM\_ECC | Encoder Internal ECC Error |
| 81 | HE\_PDH\_CRC | Encoder CRC Check for Prefix Data Failed |
| 82 | HE\_PFX\_CRC | Encoder CRC Check for Predetermined Huffman Tables Failed |
| 83 | HE\_SYM\_MAP\_ERR | Encoder encountered an unmappable symbol |
| 100 | CRYPTO\_ENC\_SEED\_EXPIRED | Engine Seed Expired for DRNG |
| 101 | CRYPTO\_ENC\_IV\_MISSING | Engine Misconfigured - missing IV |
| 102 | CRYPTO\_ENC\_INVALID\_SEQNUM | Engine KEY or COMMAND Sequence Number Failed(didn't increment by +1) |
| 103 | CRYPTO\_ENC\_INVALID\_ENGINE\_ID | Engine KEY and COMMAND Engine ID Mismatch |
| 104 | CRYPTO\_ENC\_KEY\_TLV\_CRC\_ERROR | Key TLV CRC-32 mismatch detected |
| 105 | CRYPTO\_ENC\_MAL\_CMD | Invalid combinations of Crypto CMD TLV Word 2 |
| 106 | CRYPTO\_ENC\_XTS\_LEN\_ERROR | Data must be at least 16 bytes for a XTS operation |
| 107 | CRYPTO\_ENC\_AAD\_LEN\_ERROR | AAD Length should be strictly less than Data Length |
| 110 | CRYPTO\_DEC\_TAG\_MISCOMPARE | Engine Authentication Tag failed check |
| 112 | CRYPTO\_DEC\_IV\_MISSING | Engine Misconfigured - missing IV |
| 113 | CRYPTO\_DEC\_INVALID\_SEQNUM | Engine KEY or COMMAND Sequence Number Failed(didn't increment by +1) |
| 114 | CRYPTO\_DEC\_INVALID\_ENGINE\_ID | Engine KEY and COMMAND Engine ID Mismatch |
| 115 | CRYPTO\_DEC\_KEY\_TLV\_CRC\_ERROR | Key TLV CRC-32 mismatch detected |
| 116 | CRYPTO\_DEC\_MAL\_CMD | Invalid combinations of Crypto CMD TLV Word 2 |
| 117 | CRYPTO\_DEC\_XTS\_LEN\_ERROR | Data must be at least 16 bytes for a XTS operation |
| 118 | CRYPTO\_DEC\_AAD\_LEN\_ERROR | AAD Length should be strictly less than Data Length |
| 120 | CRYPTO\_INT\_TAG\_MISCOMPARE | Engine Authentication Tag failed check |
| 121 | CRYPTO\_INT\_INVALID\_SEQNUM | Engine Misconfigured - missing IV |
| 122 | CRYPTO\_INT\_INVALID\_ENGINE\_ID | Engine KEY or COMMAND Sequence Number Failed(didn't increment by +1) |
| 123 | CRYPTO\_INT\_KEY\_TLV\_CRC\_ERROR | Key TLV CRC-32 mismatch detected |
| 130 | KME\_DAK\_INV\_KIM | KIM Entry requested in AUX\_CMD for DAK |
| 131 | KME\_DAK\_PF\_VF\_VAL\_ERR | Validation Check failed DAK for this command |
| 132 | KME\_DEK\_INV\_KIM | KIM Entry requested in AUX\_CMD for DEK |
| 133 | KME\_DEK\_PF\_VF\_VAL\_ERR | Validation Check failed DEK for this command |
| 134 | KME\_SEED\_EXPIRED | Seed Expired for KDF |
| 135 | KME\_DEK\_GCM\_TAG\_FAIL | GCM Authentication Tag failed during DEK decryption |
| 136 | KME\_DAK\_GCM\_TAG\_FAIL | GCM Authentication Tag failed during DAK decryption |
| 137 | KME\_ECC\_FAIL | ECC failure on KIM/CKV data. |
| 138 | KME\_UNENC\_KEYS\_DISABLED | Key Type 2-6 is disabled. |
| 139 | KME\_DEK\_ILLEGAL\_KEK\_USAGE | KIM marked as KEK was neither used to decrypt DEK keys nor as a DEK KDF Key. |
| 140 | KME\_DAK\_ILLEGAL\_KEK\_USAGE | KIM marked as KEK was neither used to decrypt DAK keys nor as a DAK KDF Key. |
| 150 | PREFIX\_PC\_OVERRUN\_ERROR | Prefix Program Counter Overran (illegal program construction) |
| 151 | PREFIX\_NUM\_WR\_ERROR | Prefix Program Illegal Destination Write (Illegal program construction) |
| 152 | PREFIX\_ILLEGAL\_OPCODE | Prefix Program Illegal Opcode (Illegal program construction) |
| 153 | PREFIX\_ILLEGAL\_INST | Prefix Program Illegal Instruction (Opcode not correct) |
| 155 | PREFIX\_ATTACH\_PHD\_CRC\_ERROR | CRC for Selected Predetermined Huffman Table check failed |
| 156 | PREFIX\_ATTACH\_PFD\_CRC\_ERROR | CRC for Selected Predetermined Data check failed |
| 170 | CG\_UNDEF\_FRMD\_OUT | An undefined FRMD type was received |
| 180 | ISF\_PREFIX\_ERR | The ISF detected an invalid user prefix size |
| 255 | TLVP\_BIP2\_ERROR | Parser encountered a BIP2 failure in TLV header word |

Table 6: Error Codes

#### Footer Field Processing

This section summarizes the Footer update/generation process on a block-by-block basis:



### CDDIP Command Processing

The CCDIP processing is like the validation function of the CCEIP engine, however there are some notable differences. These are:

* The Prefix Attach Engine will need to consider the actual payload data to retrieve the Prefix ID that was used. This involves parsing the incoming data, just like the decompressor to find the Prefix ID in the XP10 Frame header or XP10 Lite header.
* The Huffman Decoder will ignore the CMD.Compression fields (all the information is carried via the FRMD TLV and the actual Data TLV.
* The CCEIP Embedded CDDIP Validator do not have the Prefix Attach Block nor the EncCmp CRC Checker.

#### Inbound Streaming Fifo

* *This processing is the same for CDDIP as it is for CCEIP.*

The ISF manipulates the datapath in the following ways:

1. To support the debug functionality in the CCEIP the ISF will look at the various APB programmable registers to implement the breakpoint and single-step functionality
   * Breakpoints can be created by programming the ISF to freeze on patterns within the incoming TLVs (see Shared Support block for more details regarding capabilities).
   * Once a breakpoint has been reached, the datapath can be single stepped.
2. The ISF does the job of breaking the opaque data TLV into 2 TLVs. It will look at the CMD.Compression.Prefix field, and if the user prefix is present will divide the DATA TLV into 2 TLVs:

* User Prefix TLV (It will generate the TLV framing structure, length, etc) along w/ the data
* Data TLV Length adjustment (to subtract out the User Prefix size).

The ISF will can be presented with several different frame stackups – it manipulates them as follows:

Figure : ISF Simple Command TLV Manipulation 

1. RQE, CMD and the incoming FRMD-TLV will be copied as is to the ISF Outbound datapath.
2. The NMD TLV will be moved deeper into the stack.
3. The DATA TLV will be moved up in the stack.
4. The ISF will be the full (large) Footer TLV which will initially be populated with control fields from the FRMD-IN TLV.

d

#### CRC Checker (Encrypted/Compressed Data)

* *In the CCEIP Engine the CRC Generator/Checker function retrieves the CRC configuration (mode – CRC64/32/16 type) from the FRMD-In TLV and places the generated CRC in the Footer TLV.*

The frame data will be run through the CRC engine and compared against the EncCmpCkSum that has been copied to the Footer.EncCmpCkSum field. The polynomial used for this function is the same as the XP10 CRC64 Polynomial:

* + x^64 + x^63 + x^61 + x^59 + x^58 + x^56 + x^55 + x^52 + x^49 + x^48 + x^47 + x^46 + x^44 + x^41 + x^37 + x^36 + x^34 + x^32 + x^31 + x^28 + x^26 + x^23 + x^22 + x^19 + x^16 + x^13 + x^12 + x^10 + x^9 + x^6 + x^4 + x^3 + 1

Figure : Encrypted/Compressed Checksum Checker

* The Encrypted/Compressed CRC Checker validates the CRC-64 of the encrypted data that is in the Footer TLV.
* Errors are recorded in the Footer TLV

#### Decryption and Authentication Engine

This engine functions the same as the CCEIP Decryption and Authentication Engine for encryption but there is a modification for the authentication tag validation. The Authentication function needs to poke into the FRMD-In TLV to determine if the full digest is available or the truncated 64 bit version in the Footer. This mask needs to be applied prior to the check to properly validate the tag.

1. Authentication will check the tag present in the Footer and signal an error when it is not valid in the Completion Message.
2. In the Footer, there is a field to indicate the size of the MAC (64/128/256 bits), so the Integrity engine can properly compare the fields.

Figure : Decryption and Authentication Validation Engine TLV Manipulation



1. The Crypto Engine is used to decrypt the data and validate the authentication tag that is present in the Footer TLV.
2. Key material is presented on the KEY Interface, whereas the data and IV is present on the main datapath.
3. Any errors will be recorded in the CQE TLV.

#### Prefix Attach Engine

* *The CDDIP Prefix Attach Engine will need to look at the FRMD header to discover that a frame is XP10 or XP10 CHU compressed and then look at the XP10 Frame Header or CHU Header to determine the actual Predetermined Prefix that will be used. This is different than the Prefix Attach Engine parsing in the CCEIP (since it simply uses the CMD.Compress PrefixSelect field populated by the Prefix Engine).*

The Prefix Attach Block will examine the FRMD-In TLV field for the Mode, and then consider the XP10/CHU frame itself to discover the Prefix that was used. This engine needs to be aware of the various CHU Modes and the processing types.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **FMDIn.XXX** | **XP10.**  **Mode** | **XP10.**  **PreDefSel** | **CHU Mode.Prefix Sel** | **Action** |
| CFH Mode 4K/8K | N/A | N/A | 0 | No Action |
| CFH Mode 4K/8K | N/A | N/A | 1-63 | Insert Predetermined Prefix and Prefix TLVs |
| XP10 | ‘b00 | N/A | N/A | No Action |
| XP10 | ‘b01 | 0-63 | N/A | No Action |
| XP10 | ‘b10 | 0 | N/A | No Action |
| XP10 | ‘b10 | 1-63 | N/A | Insert Prefix TLV |
| XP10 | ‘b11 | 0 | N/A | No Action |
| XP10 | ‘b11 | 1-63 | N/A | Insert Predetermined Prefix and Prefix TLVs |

Note:

Figure : Prefix Attach TLV Manipulation



* The Prefix Attach Engine will insert the Predetermined Huffman and Predetermined Prefix TLVs.

#### Decompression (Huffman Decoder)

The decompression engine takes in the TLV stackup and will perform the decompression. The Decompressor block terminates the Predetermined Huffman and Prefix TLVs.

* The Decompressor will check the CRCs present on the Predetermined Huffman and Prefix TLVs and place an error code in the Completion TLV and halt processing if there is an error.
* The Decompressor, when it detects the Prefix TLV moves that data directly to the LZ77 History Buffer.
* The Decompressor needs to look at the FRMD-In TLV and the beginning of the actual DATA TLV in order to make a determination about how to process the data. For decompression, the CMD.Compress fields will not be used.

Figure : Huffman Decoder TLV Manipulation 

1. The Huffman Decoder terminates the Predetermined Huffman TLV.
2. The DATA TLV is transformed into an LZSYM-TLV
3. The Footer TLV will be updated with any errors that occur during processing and the # of decompressed bytes.

#### Decompression (LZ77)

* The Data TLV that the Decompressor issues is of Type Data Length Unknown (since the full decompressed length isn’t available until AFTER the frame has been decompressed.

The LZ77 Decompressor uses the Prefix Data (either the 1KB Predetermined Prefix or the 1-64KB User Prefix Data to prime the LZ77 Window, and then performs the symbol expansion of the LZ77 Symbols.

* The Huffman Decoder block will perform the MTF to PTR mapping
* The LZ77 Symbol Expander will have a peak performance of 50Gbps due to the size of the outgoing datapath (64 bits @ 800MHz).
* The XP9/XP10/Deflate Adler will be checked post decompression in this engine.

Figure : LZ77 TLV Manipulation



1. The LZ77 Decompressor will terminate the Prefix TLV.
2. The LZ77 Decompressor will transform the LZSYM TLV back into Data.
3. Appropriate CRCs will be checked and any errors propagated to the FRMD-OUT and CQE TLVs.

#### Integrity/Authentication (Crypto) Engine

The final datapath check that will be created is an integrity/authentication digest generation on the Raw Data. This is done at the end of the pipeline by a version of the Crypto engine that does not have the AES core. This authentication tag will be either SHA2 or SHA2-HMAC. The authentication tag is populated in the Footer.

Figure : Raw Integrity and Authentication Engine



* The Authentication engine consumes a KEY TLV that is presented on the KEY interface and uses the main Datapath to build an authentication tag which is placed in the Footer TLV.
* The FOOTER TLV will be updated with any errors that occur during processing.

#### RawData Checksum Checker

* This engine is the same as the CCEIP’s RawData Checksum Checker, but instead can build a Raw CRC of the decompressed, decrypted data.

This block will build a CRC over the Raw Frame Data specified in the DATA TLV and populate in the FOOTER TLV.

The polynomial used for this function is the same as the XP10 CRC64 Polynomial:

* + x^64 + x^63 + x^61 + x^59 + x^58 + x^56 + x^55 + x^52 + x^49 + x^48 + x^47 + x^46 + x^44 + x^41 + x^37 + x^36 + x^34 + x^32 + x^31 + x^28 + x^26 + x^23 + x^22 + x^19 + x^16 + x^13 + x^12 + x^10 + x^9 + x^6 + x^4 + x^3 + 1

Figure : Raw CRC Generator TLV Manipulation



* The Raw CRC checker will compute the CRC as instructed in the FRMD-IN-TLV and compare against the CRC that is in the Footer

#### Outbound Streaming Interface

The outbound streaming interface takes the data from the decompression engine and starts writing it to the RXC engine via the OSF. This block is the same as the CCEIP version, but intercepts the datapath at a different point (CCDIP = output of the Decompressor, CCEIP = output of the Crypto Engine).



* The outbound streaming interface takes the data TLV and outputs it to the RXC.
* It also generates the FRMD based on the FOOTER.
* It outputs the STATSs and CQE TLVs as well (the Footer is used to build the final error codes in the CQE TLValong wit the STATs and Completion TLVs.

#### Footer Field Processing

This section summarizes which blocks contribute to the Footer’s fields



### Error Reporting/Handling

* The list of engine error codes is maintained in 3.1.5.12.

### Debug Features

* + APB bus will be able to generate a block of data and push it through the compression and crypto engines
  + Each block can be fed data directly from the *Inbound DataStream*; additionally the *Outbound DataStream* can be fed by any of the engines (compression/crypto/etc). This is to facilitate debug as well as block level verification.
  + Including the AUX\_CMD TLV in this block of data will allow error conditions (e.g. Digest Errors, CRC errors, et) to be simulated using the scheme described in 4.1.5. As a result, each of the Completion Event Error Codes can be triggered by software, and aid in the development/debugging of the software that handles these errors.
  + Trigger-Freeze will be supported on the *Inbound DataStream.* This mechanism allows the user to provision in some sort of trap condition based on the incoming header stackup. As the block transits the system, the internal memories and state will be held so as to allow software to get in and read the state of the system via the APB bus. There is also a single-step feature that can be used to slowly step the system through and read out intermediate state. The Trigger-Freeze mode is described in Section 3.3.1 (Dataflow Control) of the Project Zipline SSB Micro Architecture Specification.

### AUX\_CMD Debug Field Usage

The AUX\_CMD has the capability of inserting errors for verification and software validation tasks.

* Production devices have an OTP bit that disables this functionality in hardware.

The AUX\_CMD.debug\_ctrl word (3.1.2.8) has all of the controls for using this debug capability. The debug functionality provided is described in Section 11.2.5 (Debug Support) of the Project Zipline SSB Micro Architecture Specification.

#### ISM

The Intra-Stage Monitor is an AXI based monitor which provides a small storage fifo to snapshot the databus.

* This engine can work non-intrusively and record the last N transactions on the bus, or generate backpressure and be lossless.
* This engine is the planned primary debug capability of the CCEIP/CDDIP engines.

#### Performance Monitors

A single Performance Monitoring engine will consolidate countable events from each of the CCEIP/CDDIP sub-blocks into a smaller number of counters

* Each subblock will output event strobes
* There will be a series of counters along with a programmable counter profile register per counter to enable that counter to record the selected event. A list of all of the countable events for the CCEIP/CDDIP is available in the Section 7.2.2 (Countable Events) of the Project Zipline SSB Micro Architecture Specification
* These counters have a snapshot mode so as to allow a coherent set of counters to be recorded. The snapshot mode is described in Section 7.2.1 (Controls) of the Project Zipline SSB Micro Architecture Specification.
* These counters will be based on the TRACE bit, so entire commands can choose or not choose to be counted via these counters.

# Datapath Bandwidth Analysis

The worst case header stack up with minimum frame data size is computed below to ensure the overhead of the TLVs do not impinge on line-rate performance.

The CCEIP/CDDIP Pipelines need to carry a 2KB frame at 25Gbps line rate:

The internal datapath of the pipelines can carry 8 bytes per clock. This datapath will need to accommodate the frame data along with the complete TLV and associated metadata; the worst case is 494 beats of data ( at a rate of 1 beat per clock).

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Items** | **Size (in bytes)** | **Beats** |
| 0 | RQE Header | 16 | 2 |
| 1 | CMD | 32 | 4 |
| 2 | FRMD In | 32 | 8 |
| 3 | Predetermined Huffman TLV | X | 67 |
| 4 | Predetermined Prefix TLV | 1 TLV  128 Databeats  1 CRC | 130 |
| 5 | Frame Data | 1 TLV header  256 Datawords | 257 |
| 6 | Footer TLV | 1 TLV  1 Ctrl Info  2 CRC (2x64 bit)  4 GUID  1 IV  8 SHAx2 | 20 |
| 7 | NMD TLV | 8 Bytes of Metadata | 2 |
| 8 | Stats TLV |  | 2 |
| 9 | CQE TLV |  | 2 |
| Total | | | 494 |