

# **General PCI Express AS Transfer Protocol Proposal**

**Bob Davis**

**Network Appliance**

Original 21 November 2002

**Revised 5 January 2003**

# Proposal Outline

- Problem Statement
- PCI Express AS Rev 0.8 coverage
- Market Segments Covered
- General Payload Requirements
- Proposal Overview
- Elements of the Payload Header
- Next Steps
- Conclusions

# Problem Statement

- Currently AS does nothing other than configure itself!
- Need basic mechanism to tunnel PCI(\*), and other endpoint architecture, through AS Fabric
- Need basic mechanism for **Native** AS usage.
- AS is Higher Level Connectivity for PCI(\*) and OPA
  - OPA – Other peoples Architecture
- Need Growth Path for PCI and NNPCI (Non-Native-PCI) systems. No legacy constraints. ie. **64 bit clean**
- Expected Lifetime = 10 + years
  - ISA lasted 15+ years
  - Plan for System growth factors
  - Plan for Application Growth factors
- Shall Not Impact Header – Proposed Header Changes independent
  - Exception being Encryption enable bit and use of turnpool as key index
- Shall Be TRANSPARENT to Switch for normal transfers
  - Exception - Switch is Target (Configuration Operation)
  - Exception - Switch is Source (Event/Interrupt operation)
- Should support Secure Packet Delivery (encryption)

# **Why PCI Express AS rev 0.8 does not solve problem**

- This problem has not yet been addressed except by this proposal and the SLS proposal
- A transfer mechanism must be in 1.0
- Current PEI-4 for Configuration has no method for validation or protection
- Validation and Security not considered
- Large Systems not planned/allowed for
- Multiple Heterogeneous Peer-to-Peer Systems solution required

# Market Segment Served

- Class 1 through Class 6 Products
  - See earlier product classifications
- Future Native AS based products
- No market segment is served now!
  - No existing market constraints
- Inclusion of capability of Non-Native PCI transfer Mechanism
- Server and Storage Systems markets
- With TBD very short header – Telecom Markets

# General Packet Requirements

- Simple Read/Write Protocol
- DMA Transfer Protocol
- Source/Destination device identification
- Priority required in packet data for endpoint use
- Error Detection CRC coverage
- Security Key Validation
- Available to support a Coherency Protocol
- Time Stamps Capability
- Capability of supporting encrypted packets
- Flexible extension for Company Usage
- AS Address/ Name Space Domain
- Extensible for Future Use
- Command and Sequence Number

# General Packet Requirements (Cont)

- Definition to allow
  - Basic Low Overhead Data Transfer
  - Optional Additional Features
- 64 Bit Clean definitions
- SAR support of large data transfers
- Encryption capability
- Include Version number for future developments
- All Transfers Packet Based Write Only
- Support Longer Transfers of:
  - DataSize(31:0) with needed data alignment and overhead
- Shortest transfer is 2 octlets – 16 bytes including 4 data bytes.

# General Packet Transfer Proposal

- This proposal is for a General PACKET Transfer Definition
- This is Packet payload includes:
  - Command Structure for Payload – 1 Doublet (16 bits)
  - Tag Field/Immediate Data Field – 1 Quadlet (32 bits)
  - Payload Size determined by Data Size Field – up to  $2^{32}$  Bytes
  - OPTIONAL Features for Reliability
    - Source Identification by EUI and Offset – 128 bit Aligned
    - Destination Identification by EUI and Offset – 128 bit Aligned
    - 32 Bit ECRC for Packet Reliability
    - Validation Key – 64 bit aligned
    - Time Stamp – PTP/NTP – 64 bit aligned
- Only Command and Tag field Required
  - All other fields are controlled by Command
- Supports FAST and Robust transfers
  - Fastest Transfer – 2 Octlets (with Route header) up to 4 bytes of Immediate Data
  - Longest Transfer – Determined by DataSize and Header length.
- Support SAR – Segmentation And Reassembly - functionality
- Support encrypted packets with payloads using AES

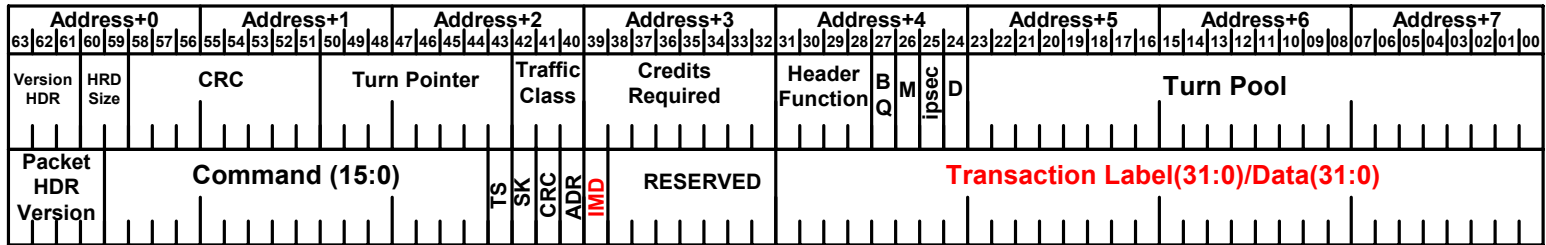
# Basic Payload

- Routing Header is assumed to be present.
- Added Command Field and Label Field
- Encapsulated incoming data
- Optional features
  - Address Block
    - Describe Destination address
    - Describe Source Address
    - Describe Data size and alignment
  - Security Key for validation
  - Time Stamp
    - IEEE Std 1588-2002
    - IETF RFC-2030
  - ECRC Dword (doublet) appended to packet
- SAR Functionality

# Additional Requirement

- Credit limit be extended to handle
  - $(\text{DataSize}(31:0)/64) + 3$  (overhead and Route header)
  - 67 needed for full format
  - Possibly 1 for future use – by committee definition
- Support PEI-4 definition here
  - Capable of added Security
  - Capable of Validation by Key
  - Capable of Source and Destination Validation
  - Support moving Configuration Cycle to Common Configuration Address Space
- Maintain 64 Bit, octlet, alignment where possible

# Short Basic Transfer with RH



Simplest Transfer for 4 bytes - Efficiency = 25%

With 16 Byte transfer - Efficiency = 50%

With 256 Byte transfer - Efficiency = 94.1%

With 4096 Byte Transfer – Efficiency = 99.6%

Immediate Data Bit (IMD) Set = Data in Label area

No CRC Required.

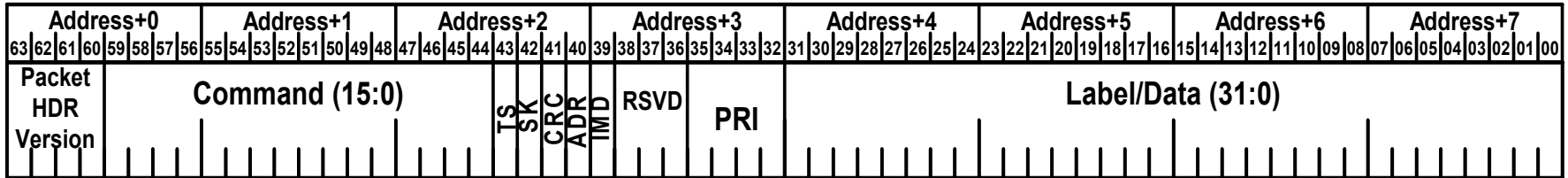
Route header is either original or newly proposed

Version – New proposal shown.

# Basic Command Field

- Defines remainder of packet
- Defines Optional function included
  - Time Stamp
  - Security Key
  - Address block
  - CRC
  - Immediate Data
- Includes Priority Field
- Supports Very Short Packets
  - High Efficiency
- Support Full Function Packets
  - Full Functionality and security

# Command octlet



Fields:

Bits 63:60 -> Packet Header Version – allow future change

Bits 59:44 -> Command Field

Bit 43 -> Time Stamp enable bit

Bit 42 -> Security/Access Key enable bit

Bit 41 -> ECRC enable bit – adds End-to-End CRC32 field

Bit 40 -> Address Field enable

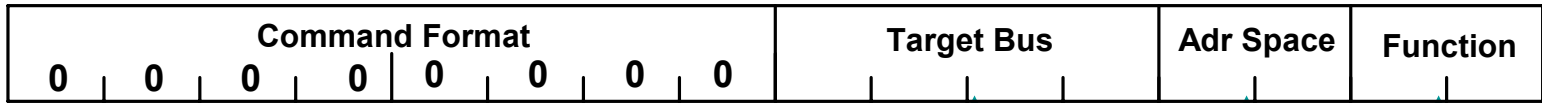
Bit 39 -> Immediate Data – Data included in place of Label

Bits 38:36 -> Reserved for future use

Bits 35:32 -> Priority field for packet at terminus

Bits 31:0 -> Data Label field or Immediate Data Field

# Command Field (0000 definition)



**Target Bus**

- 0000 = Message
- 0001 = PCI Express
- 0010 = PCI, PCI-X\*
- 0011 – 1111 = TBD

**Address Space**

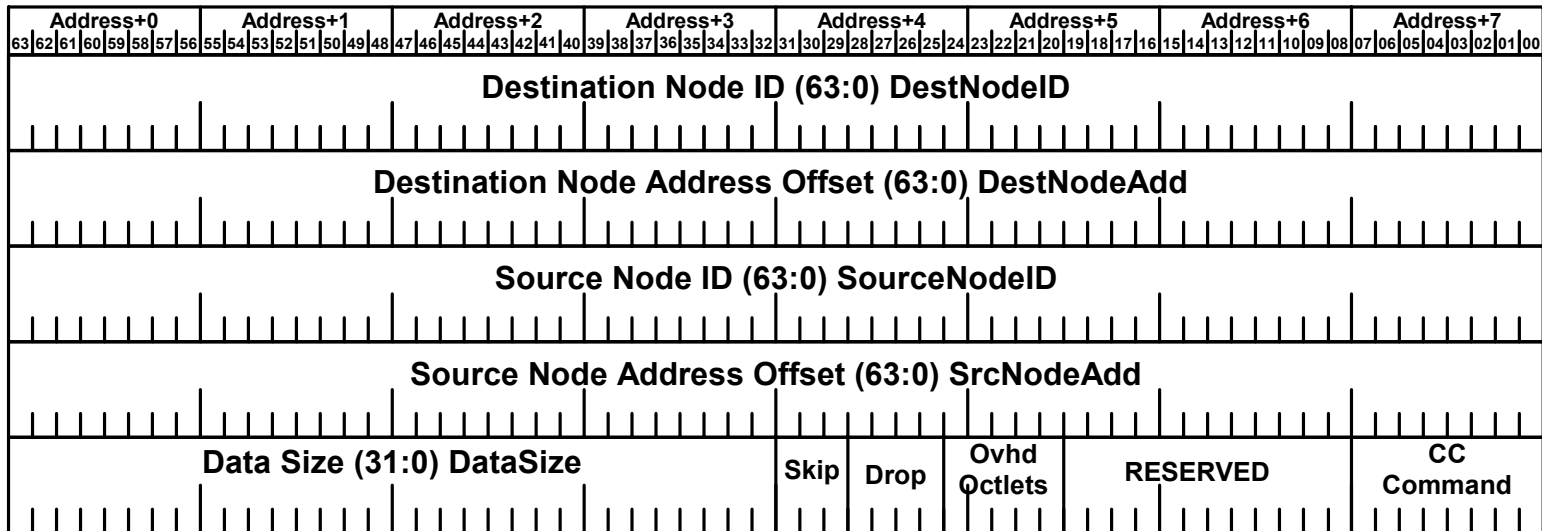
- 00 = Memory Access
- 01 = Configuration Cycle
- 10 = I/O Access
- 11 = Event Space

**Address Space and Function Code  
Are for PCI Bus – Target = 0010**

**Function Code**

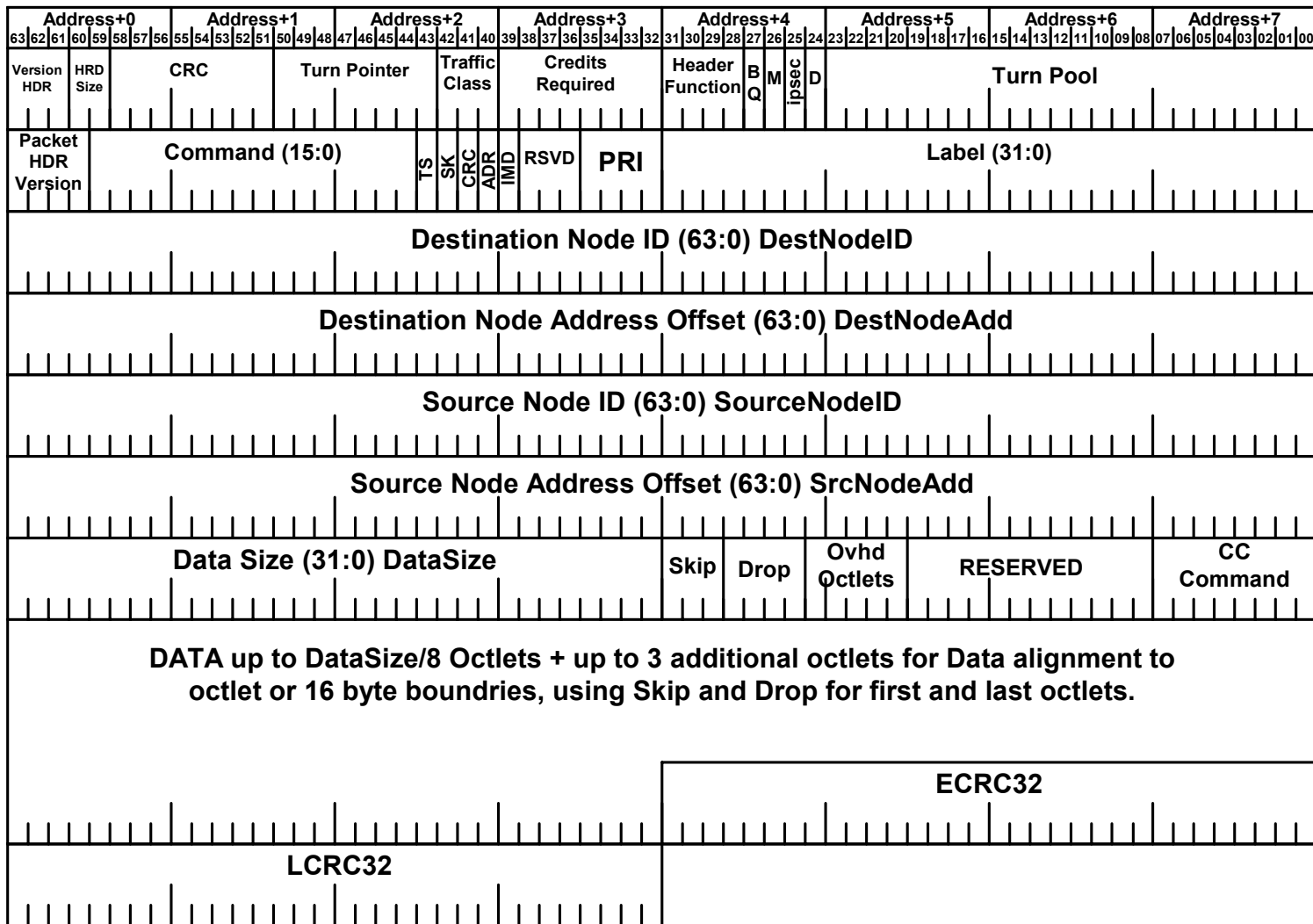
- 00 = Write
- 01 = Read
- 10 = Response
- 11 = Event

# Address Block (if enabled)



- Complete Address for both Destination NodeID and offset (node address)
- Complete Address for both Source NodeID and offset (node address)
- Data Size in bytes up to  $2^{32}$  bytes
- Added octlets include all bytes not in payload
  - This block add 5 Octlets = 10 Quadlets/Dwords = 40 bytes
- Added Cache Coherency Command Field - TBD
  - All zero is not used.
- Skip is the number of bytes/octets to skip in first octlet transferred before accepting data.
- Drop is the trailing number of bytes/octets to drop in last octlet transferred
  - Example – 2 bytes transferred with starting address of xxx007 would be conveyed by DataSize of 2 and a Skip of 6 and Drop of 6.
  - Data Transferred is 16 bytes- 2 octlets - for 2 bytes delivered.
- 12 bits of space reserved for future use.

# PCI transfer with Address and CRC



Basic PCI type transfer and probable SAR Candidate





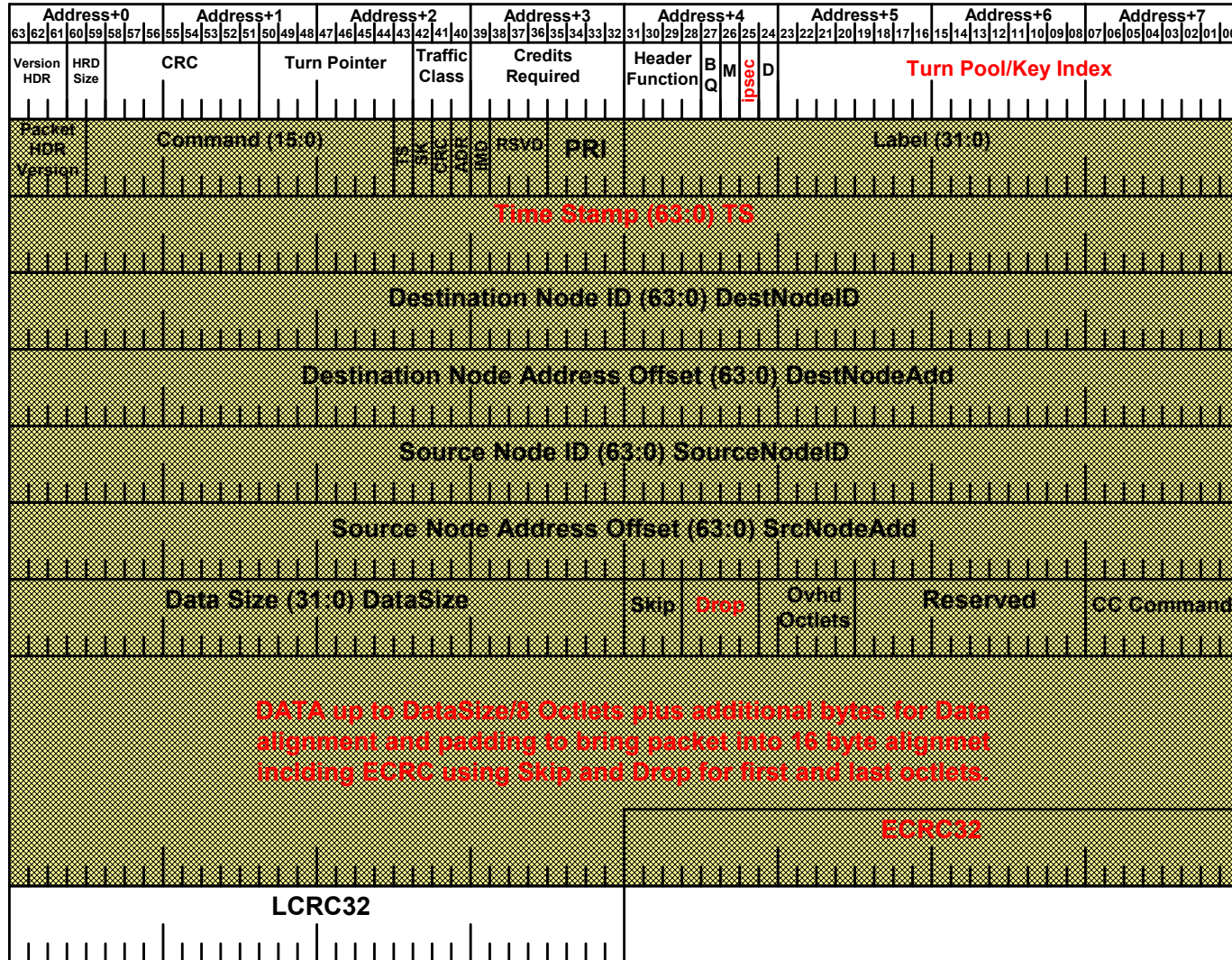
# Data Field

- 0 to  $2^{32}$  bytes
- Initial data to be ~ 256 bytes per vendors.
- Keep 64 bit octlet alignment
  - 2 byte transfer may require 4 quadlet(Dword)  
= 2 octlets depending on data alignment
- Length Defined in Address Field if used
- Longer transfer packet will be supported in future switches

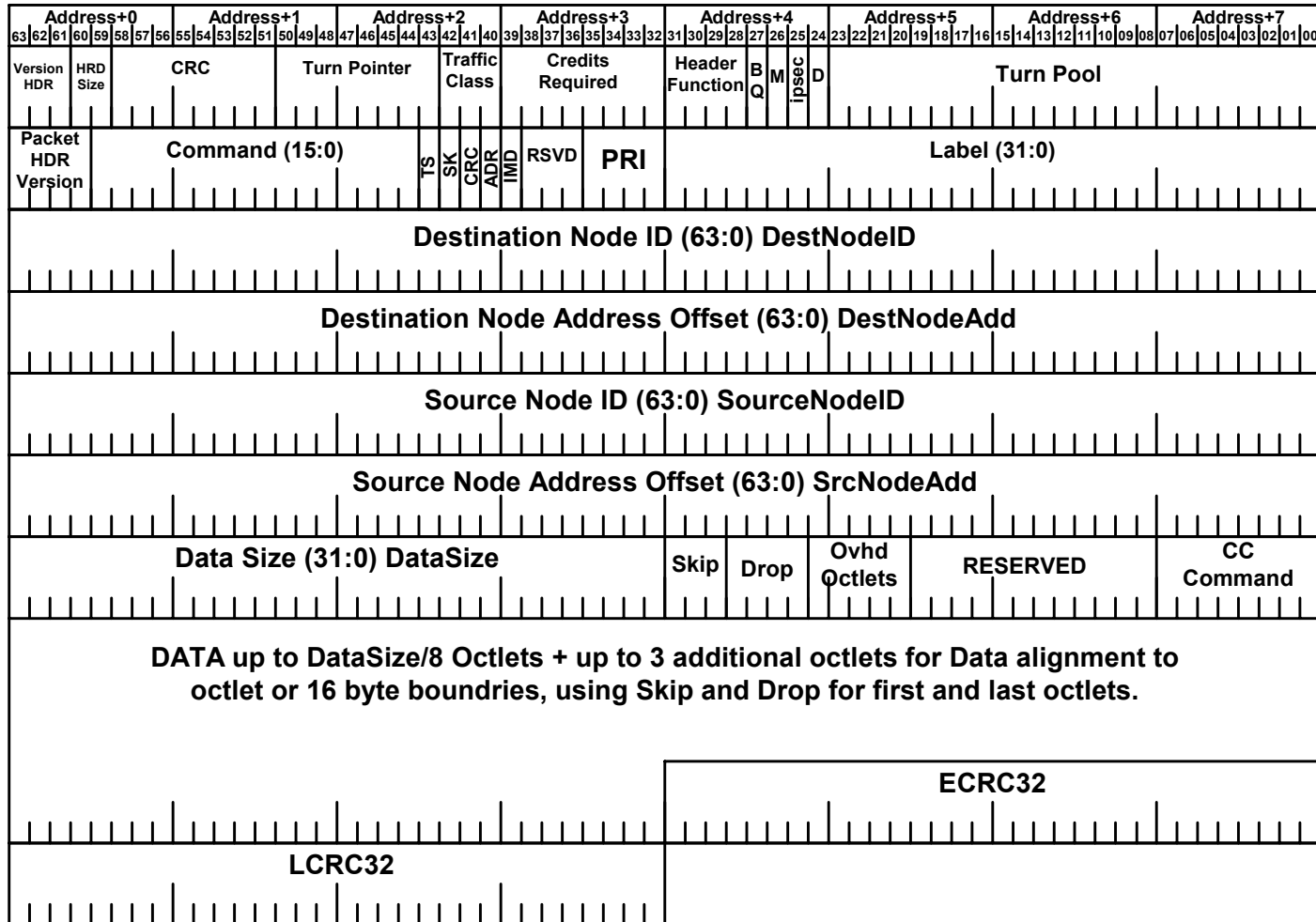
# Encrypted Data Transfer

- Used with support from Route Header
  - Ipsec Bit set in route header indicates payload is encrypted from start of Command octlet to end of ECRC.
- LCRC is not encrypted
- Route turnpool serves as index into encryption key index.
  - Long Routing header use:
    - Hash of long header as Key index
    - First 24 bits of Turn Pool is used as Key Index
  - Multipath transfers enhances security
- Additional Support in Packet address block to ensure:
  - Proper 16 byte alignment for 128 bit cipher blocks
  - Added definitions in Drop field to support alignment
  - Time Stamp/PTP support Initial Vector requirements

# Encrypted Data Transfer



# Proposal for SAR Packet



Command Field determines function in SAR transfer

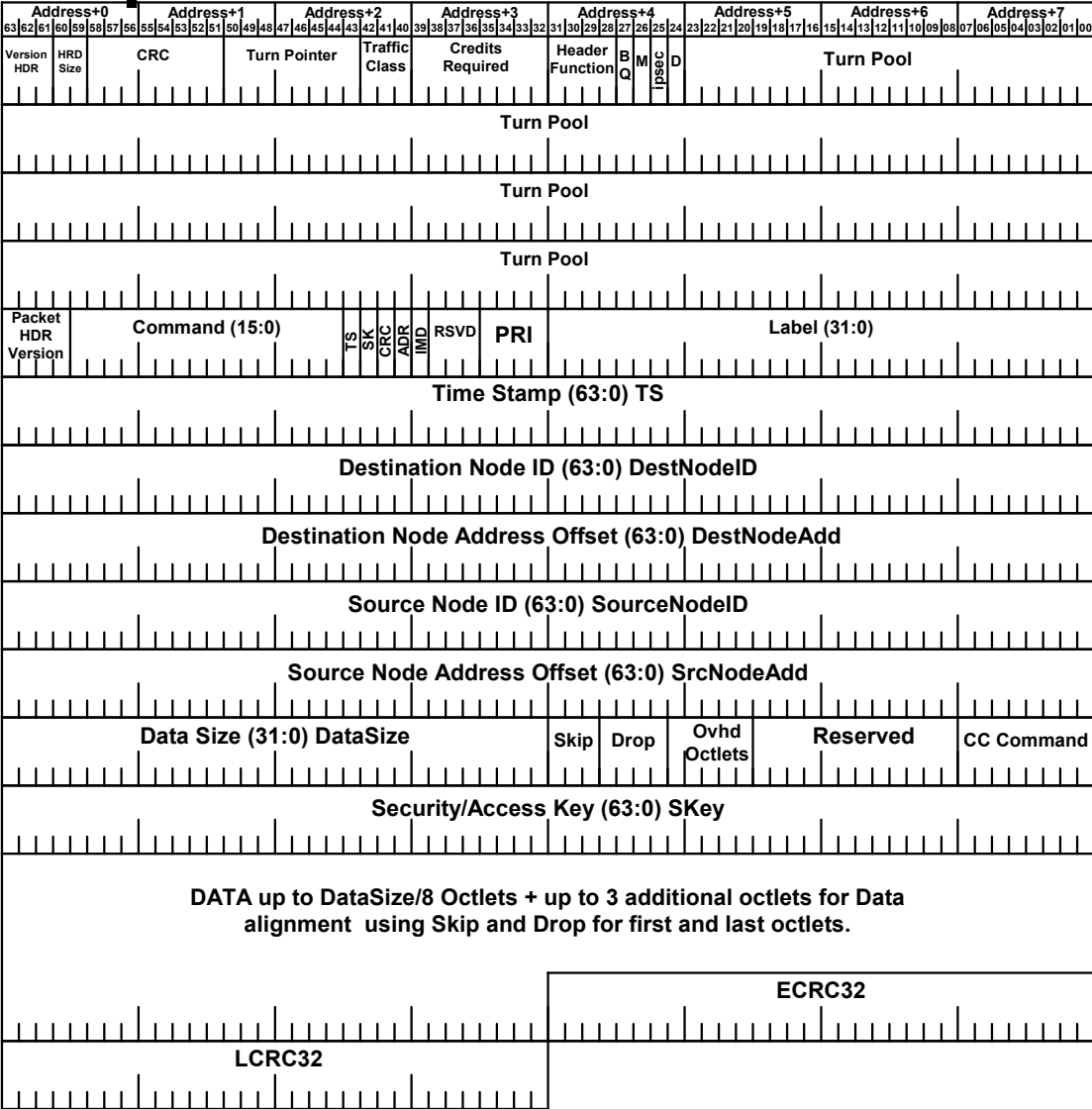
# SAR Command Usage

- This general mechanism can use the DestNodeAdd and the SrcNodeAdd in two ways:
  1. As pointers defining the queue pair participating in this operation.
  2. As pointers, through a Protection Translation Table, if one exists, or directly to the appropriate memory locations in the respective endpoints. This would require that the addresses change to properly deposit the data in the proper place.
- The Label(31:1) is incremented with each succeeding data packet.
- With the addition of the time stamp, this set of SAR transactions can be encrypted for transfer.
- The Command field guides the setup and progress for the SAR operation
  1. Command Field (15:0) = 01xx<sub>h</sub> for first transfer in a SAR operation
  2. Command Field (15:0) = 02xx<sub>h</sub> for middle transfers in a SAR operation
  3. Command Field (15:0) = 03xx<sub>h</sub> for last transfer in a SAR operation
  4. Command Field (15:0) = 04xx<sub>h</sub> for setup of a SAR operation where the addresses and length are for the overall data block to be SAR'ed

# CRC (if enabled)

- Optional per Command Field
- If CRC bit is set the system appends a 32 bit CRC32 per definition in Base specification
- LCRC follows this optional ECRC
- Coverage is Command Field through Payload Data.

# Complete Transfer Package



# Next Steps

- Following presentation at Chandler F2F
- Seek input from Group
  - Incorporate changes
- Meetings for discussion
  - Telephone
  - Next Face To Face
- Develop State Machine diagrams for “0” function
- Create appropriate Text
- Target inclusion in Ver 0.9 of AS Spec
- Complete by Release Candidate 1.0

# Conclusions

- General Encapsulation Payload Header
- Provides for Short Encapsulation overhead
- Provides for One set of fuller functionality
- Provides for Many other sets of headers
  - This is set 0 in the Header
  - Versions 1 to 15 not defined - RFU