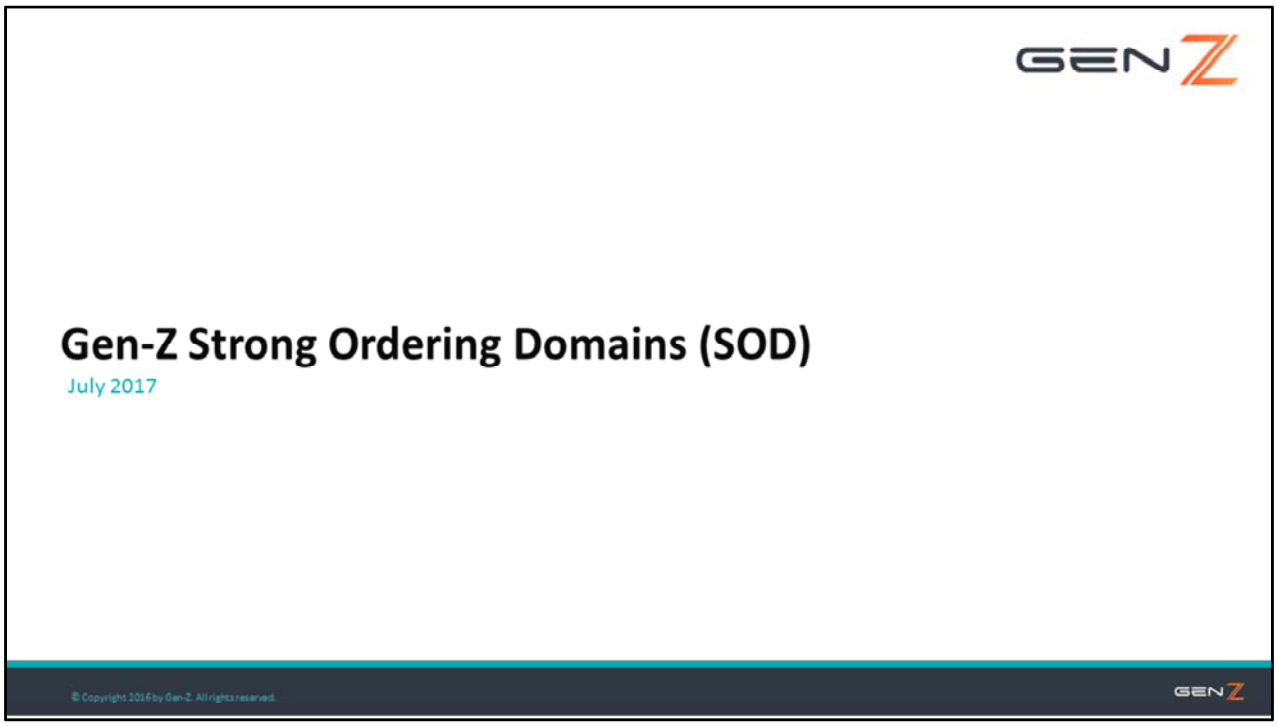


This presentation covers Gen-Z Strong Ordering Domains (SODs). Most Gen-Z solutions use datagram communications which enables packets to flow across multiple paths, apply adaptive routing, etc. SODs are used by a subset of applications that require packets to be received in the same order that they were transmitted.

Disclaimer

This document is provided 'as is' with no warranties whatsoever, including any warranty of merchantability, noninfringement, fitness for any particular purpose, or any warranty otherwise arising out of any proposal, specification, or sample. Gen-Z Consortium disclaims all liability for infringement of proprietary rights, relating to use of information in this document. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted herein.

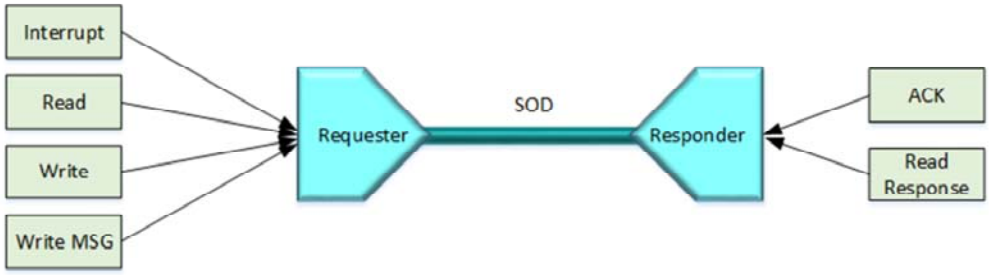
Gen-Z is a trademark or registered trademark of the Gen-Z Consortium.

All other product names are trademarks, registered trademarks, or servicemarks of their respective owners.

All material is subject to change at any time at the discretion of the Gen-Z Consortium

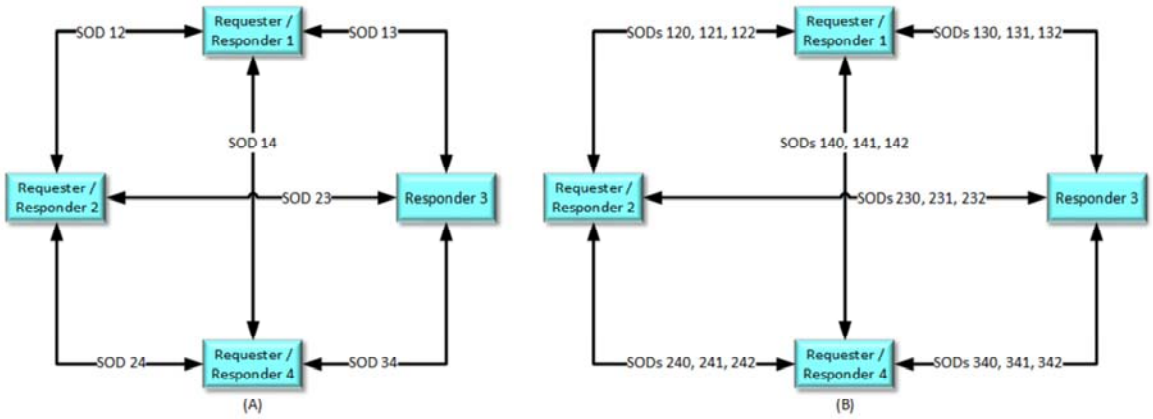
http://genzconsortium.org/

Strong Ordering Domain (SOD)



- SOD is a logical construct used to delineate individual packet flow, provide Reliable Delivery, and enforce strong packet ordering.
- Conceptually, a SOD is a pipe through datagram request and response packets are exchanged in the order posed
  - Packets are multiplexed over a given SOD which ensures packets are delivered in the order posted
- Each SOD uses a separate sequence number space in place of a Tag

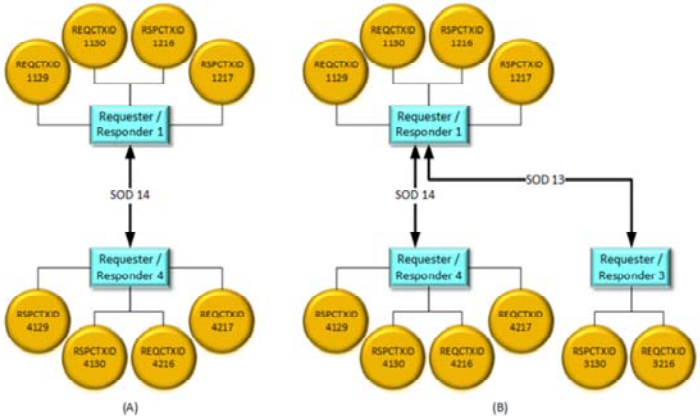
Multiple SODs



- Components can support multiple SODs between any pair of components
  - Multiple SODs enable differentiated services, enable multiple paths to be utilized, enable resiliency, etc.

GEN Z

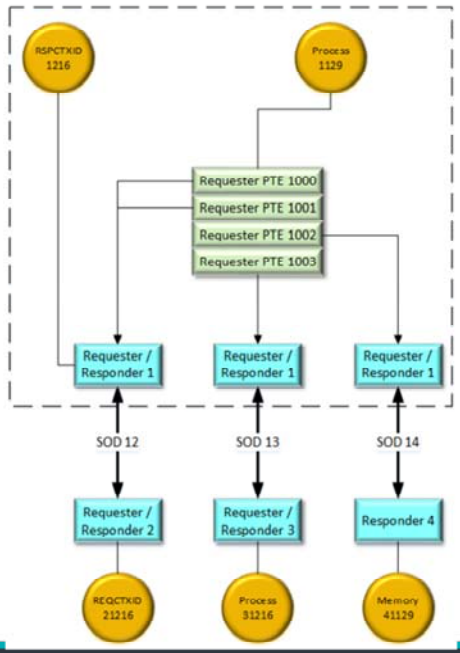
SODs and Context Identifiers



- Applications that use Contexts to delineate application instances can use SODs
- Contexts may communicate through a single SOD (A) or are cross multiple SODs (B)
  - Software is responsible for enforcing ordering when posting requests on multiple SODs (e.g., a fence bit)

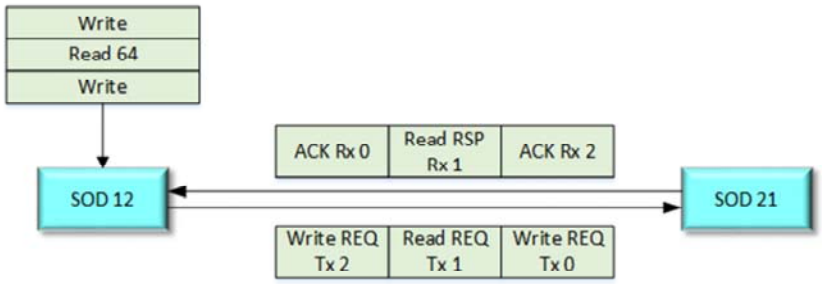
Generating SOD Request Packets

- SOD request packets can be generated just as datagram request packets through a Requester ZMMU
  - Page Attributes within the Requester PTE entry indicate that a SOD Read or SOD Write is to be generated
  - Each page is associated with a single Responder
  - Requester can use the Traffic Class or other means to select one of N SODs per communicating Responder
- SOD request packets can be generated on behalf of a Context
  - Further, request packets generated through the Requester ZMMU or through a Context can be interleaved across the same SOD



GEN Z

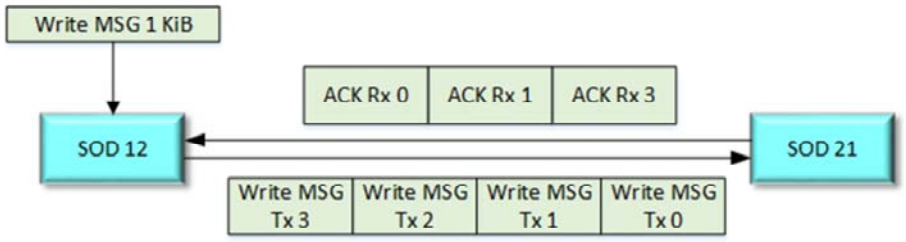
SOD Request-Response Packet Flow



SOD request packets are transmitted in the order posted using a transmit (Tx) sequence number to uniquely identify each request packet.

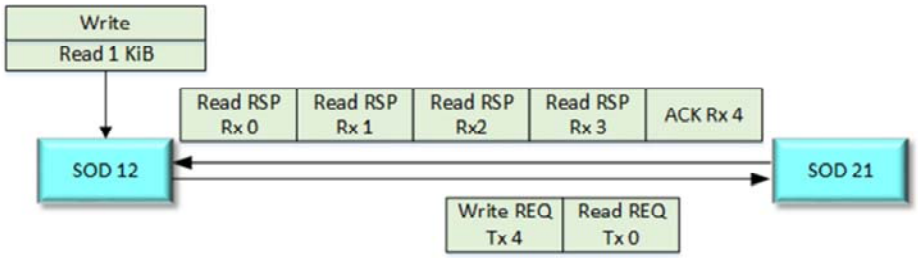
- The first write is assigned transmit sequence number 0 (Tx 0), the read Tx 1, and the last write Tx 2.
- The Responder generates a unique response packet that acknowledges all request packets successfully executed at the time the response packet was generated.
- In this example, the Responder transmits ACK Rx 0 that corresponds to write Tx 0, a read response Rx 1 that corresponds to read Tx 1, and ACK Rx 2 that corresponds to write Tx 2.

Multi-packet Write MSG



- Multi-packet operations such as a Write MSG are transmitted as a contiguous sequence of packets across a SOD
- This simplifies payload placement and determining when the operation as completed.
  - This example also illustrates that response packets are cumulative acknowledgements, i.e., ACK Rx 3 acknowledges all request packets Tx 0 through Tx 3, enabling a Responder to opportunistically reduce the number of acknowledgment packets generated.

SOD Read Request

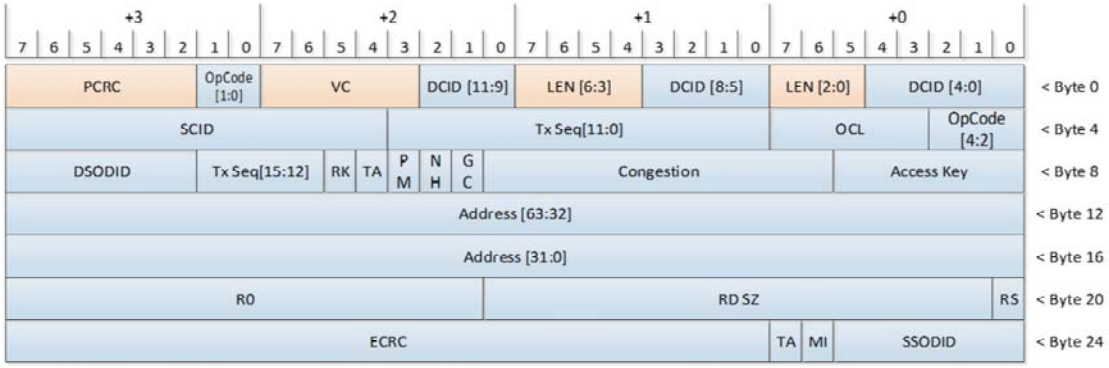


- If a read request is larger than Max_Packet_Payload, then the next transmit sequence number is incremented by N to account for the number of corresponding read response packets.
- Figure illustrates a 1 KiB read request followed by a write request.
  - Assuming a Max_Packet_Payload of 256 bytes, the Responder generates 4 read response packets.
  - The Requester assigns the read request packet Tx 0 and the subsequent write request packet Tx 4.
  - The Responder validates the LDM Read request packet, calculates the number of requisite response packets, and sets its next expected sequence number to Tx 4.
  - Advancing the Tx sequence number simplifies response packet generation and response packet payload placement.

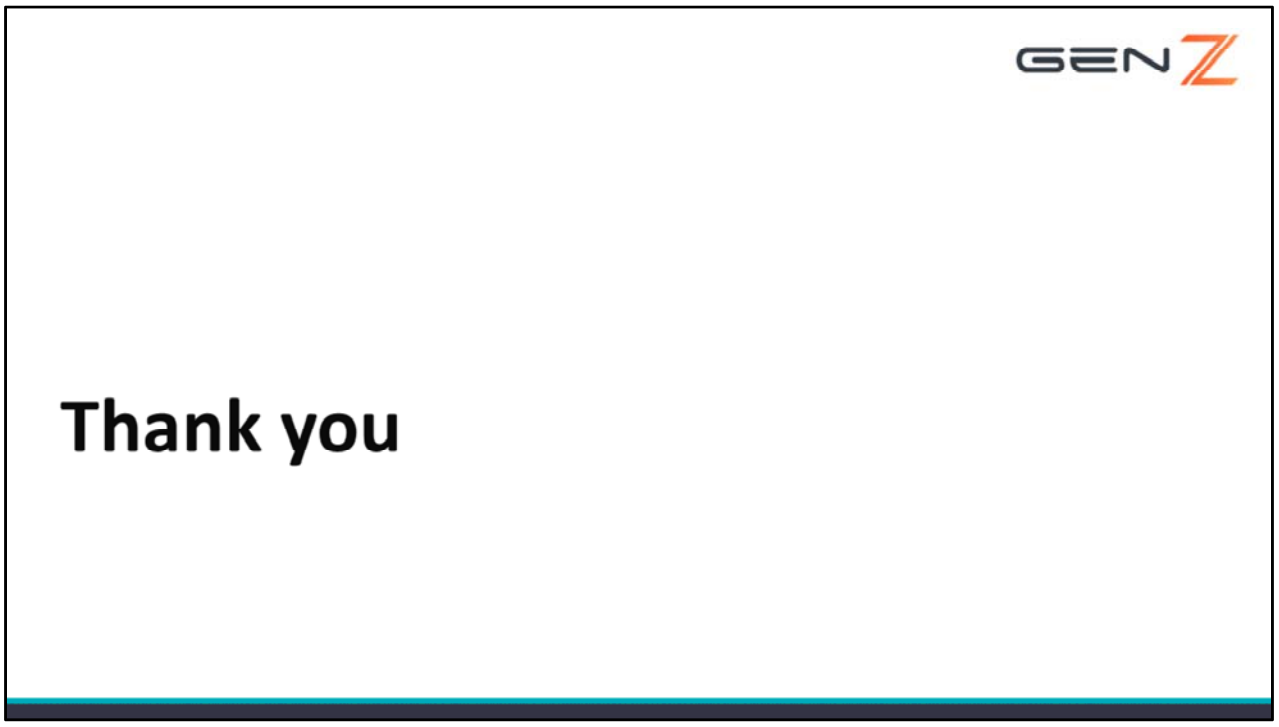
Supported SOD Operations

- SOD Read Request—read up to 2^{24} bytes of data starting at the specified
- SOD Read Response—return read data
- SOD Write—write or write persistent data starting at the specified address
- SOD Write MSG—write a message to the indicated Context
- SOD Standalone Acknowledgment—acknowledge success or communicate an error
- SOD Encapsulation—Encapsulate a non-SOD packet. Enables SODs to support any Unicast Gen-Z operation without requiring a SOD analog to be specified
- SOD Interrupt—Deliver a Gen-Z native interrupt
- SOD Sync—If a request was aborted, this operation is used to synchronize the sequence numbers to enable the two components to continue without requiring the SOD to be torn down and rebuilt

SOD Read Packet Format



- DSODID and SSODID identify the destination and source SODs
  - Up to 128 SODs can be established between component pairs
  - Applications multiple request and response packets across these SODs
- Per SOD sequence number space is sufficient to maximize bandwidth through a single SOD



This concludes this presentation. Thank you.