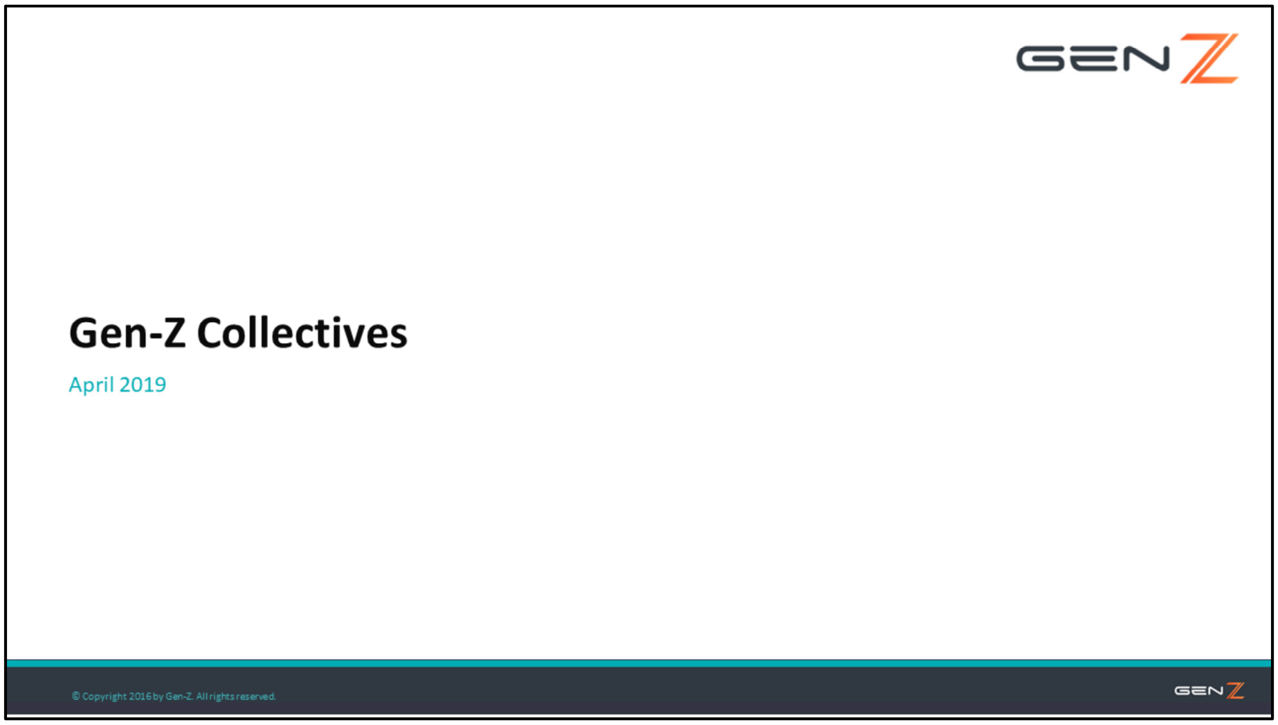
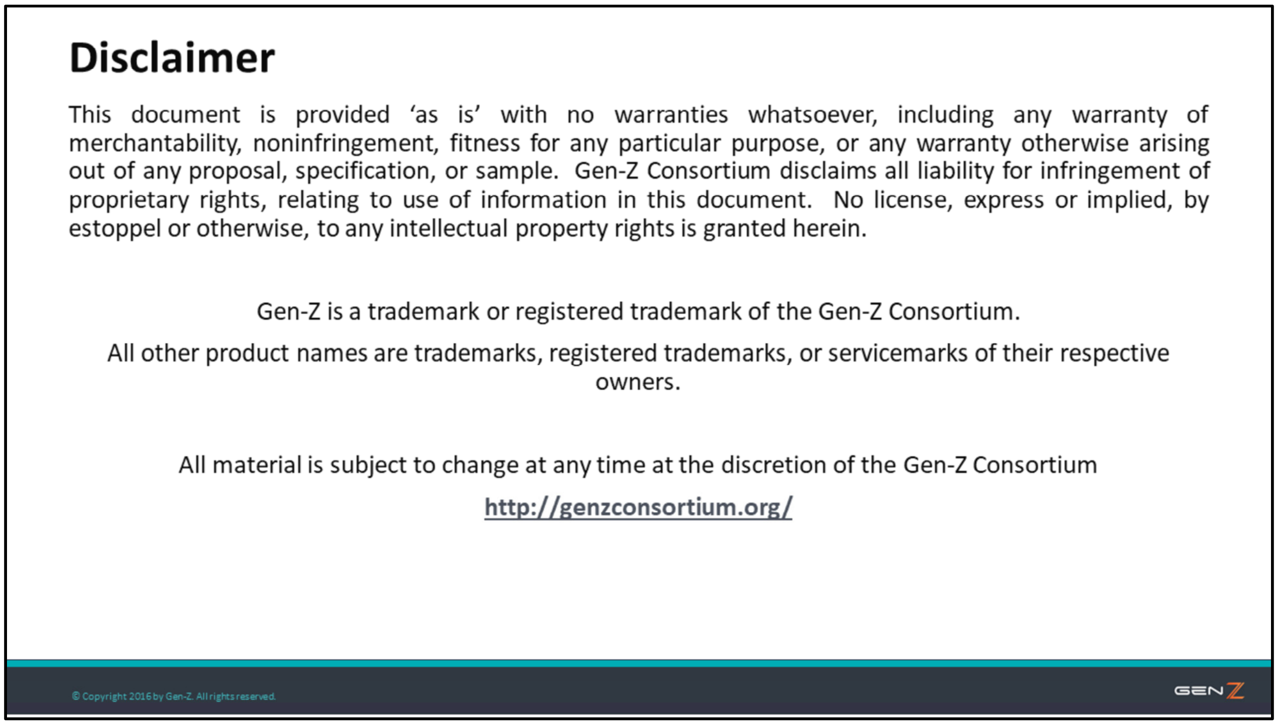


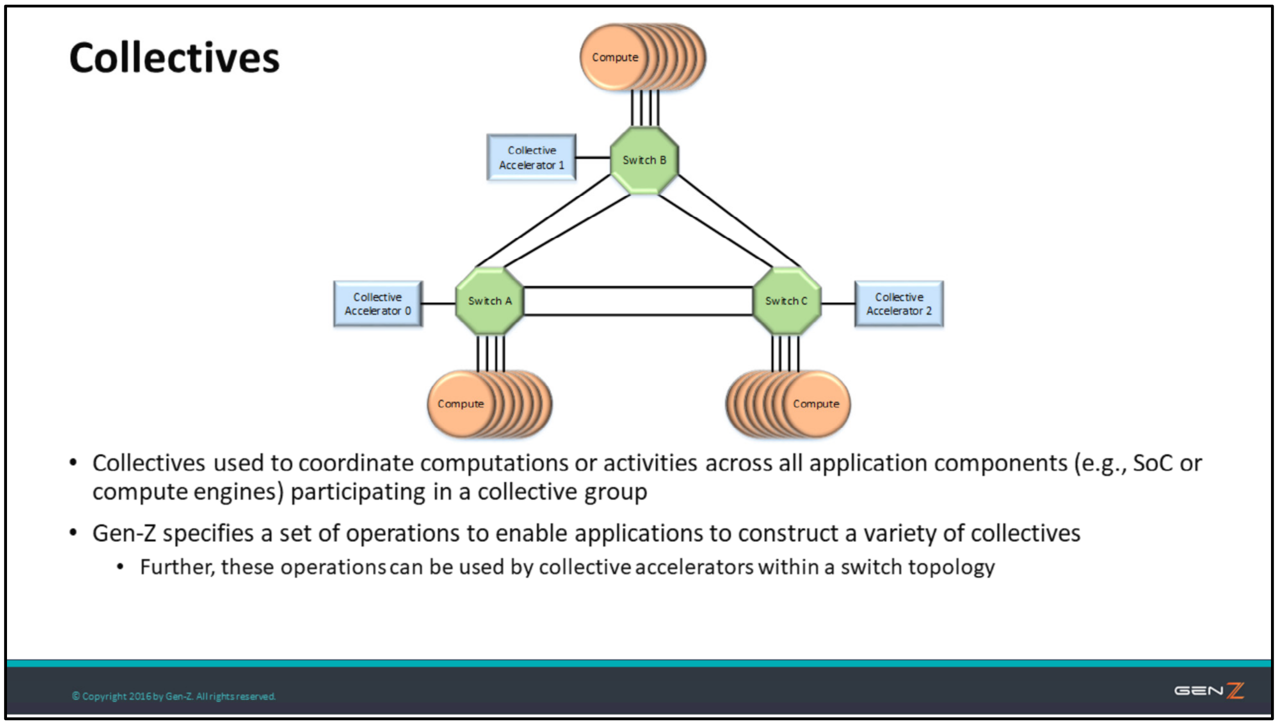
Gen-Z Collectives

April 2019

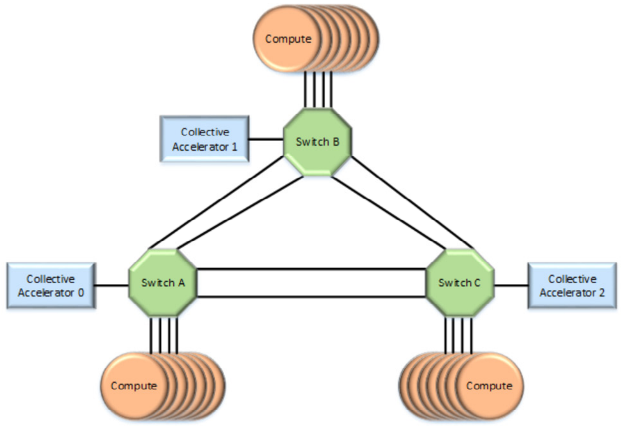
This presentation covers Gen-Z Collectives support and operations.



Disclaimer

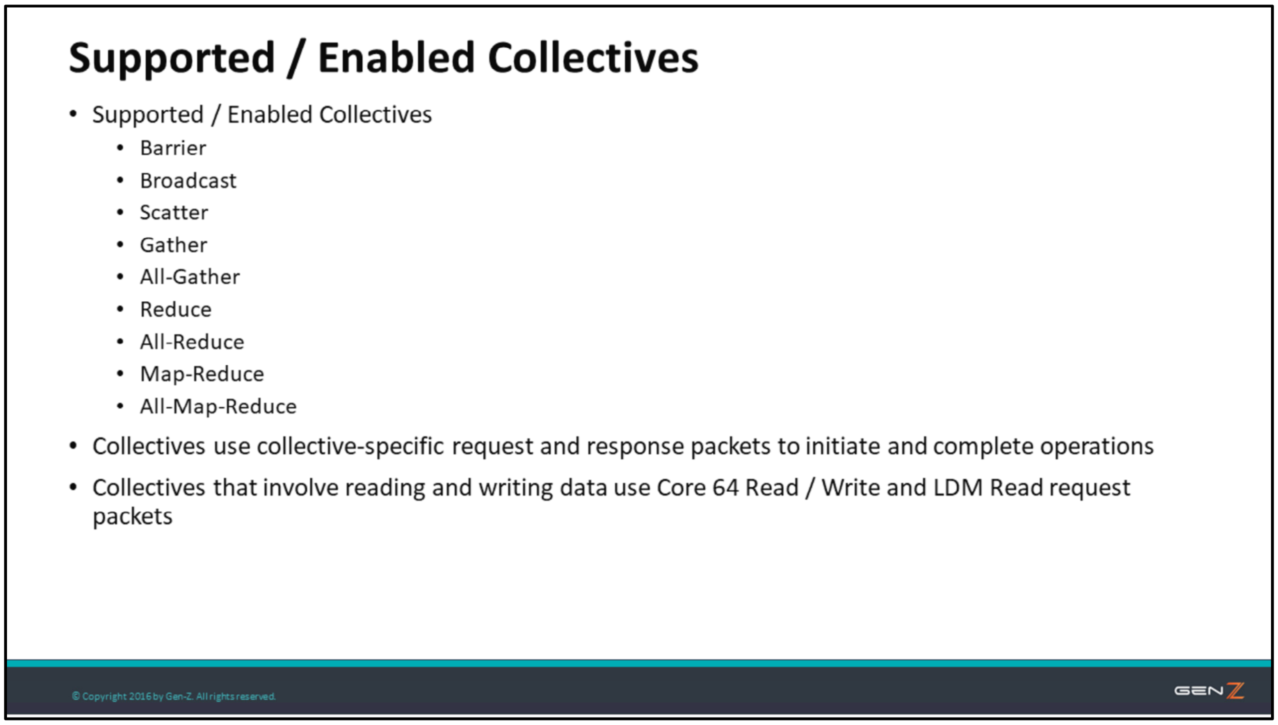


Collectives



- Collectives used to coordinate computations or activities across all application components (e.g., SoC or compute engines) participating in a collective group
- Gen-Z specifies a set of operations to enable applications to construct a variety of collectives
  - Further, these operations can be used by collective accelerators within a switch topology

Collectives are use in a wide-range of messaging applications, e.g., numerous MPI and shared memory applications use collectives. Collectives are often implemented in application or middleware software. They are used to coordinate computations or activities across a distributed application. Gen-Z specifies a set operations to enable applications to support a variety of collective operations. Further, these operations can be off-loaded to collective accelerators within a switch topology to improve application performance, reduce fabric load, and simplify implementations. A collective accelerator is a processing engine that is directly-attached to the switch topology. For example, in the above figure, one collective accelerator is provisioned per switch.

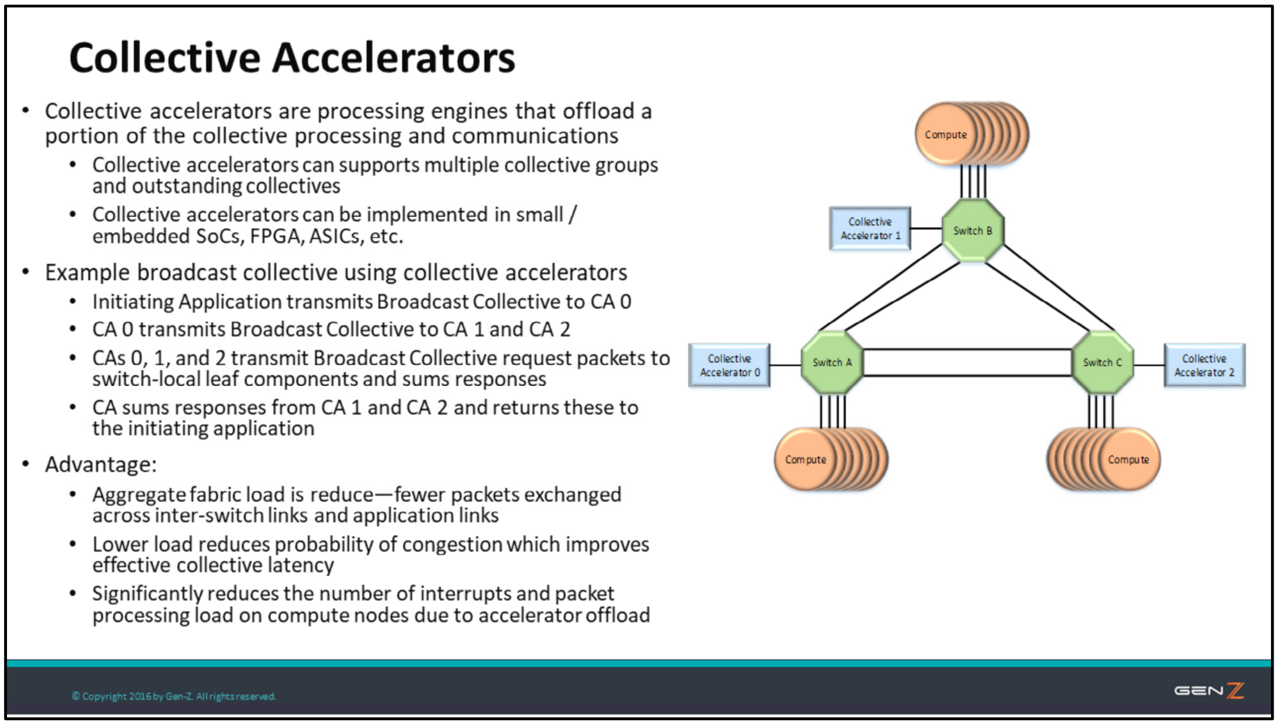


Supported / Enabled Collectives

- Supported / Enabled Collectives
    - Barrier
    - Broadcast
    - Scatter
    - Gather
    - All-Gather
    - Reduce
    - All-Reduce
    - Map-Reduce
    - All-Map-Reduce
- Collectives use collective-specific request and response packets to initiate and complete operations
- Collectives that involve reading and writing data use Core 64 Read / Write and LDM Read request packets

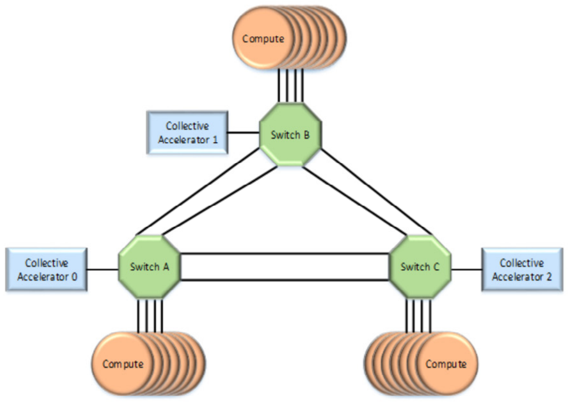
Gen-Z supports a variety of collectives.   These can be implemented using collective request and response packets or in conjunction with Gen-Z read, write, and LDM (large data movement) read request packets.

An All-to-All collective can be implemented as a Scatter plus All-Gather collective. This sequence should provide equivalent performance.

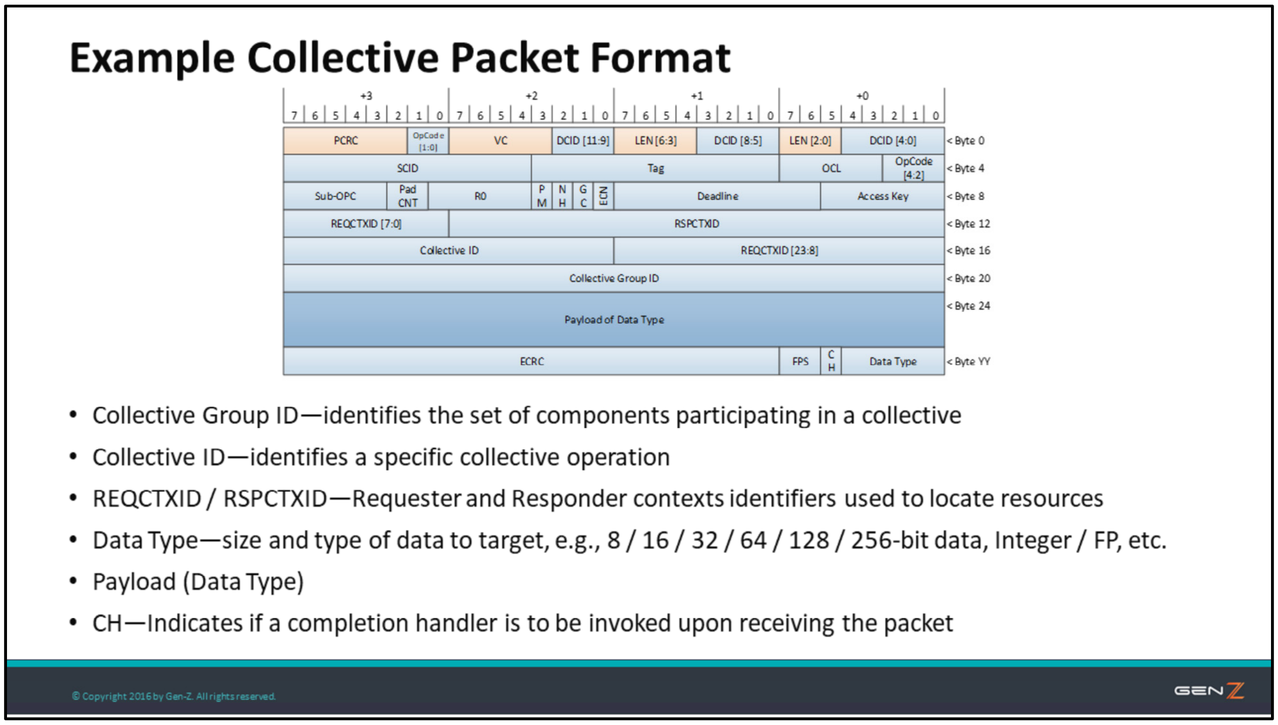


Collective Accelerators

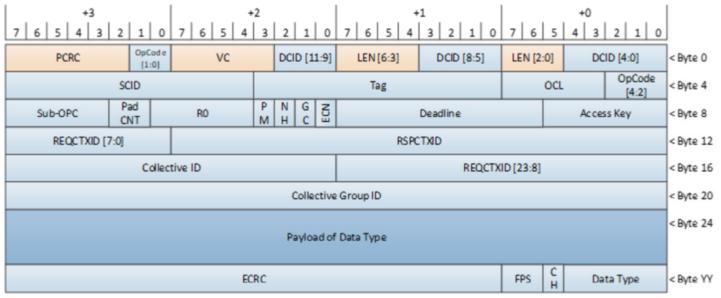
- Collective accelerators are processing engines that offload a portion of the collective processing and communications
  - Collective accelerators can supports multiple collective groups and outstanding collectives
  - Collective accelerators can be implemented in small / embedded SoCs, FPGA, ASICs, etc.
- Example broadcast collective using collective accelerators
  - Initiating Application transmits Broadcast Collective to CA 0
  - CA 0 transmits Broadcast Collective to CA 1 and CA 2
  - CAs 0, 1, and 2 transmit Broadcast Collective request packets to switch-local leaf components and sums responses
  - CA sums responses from CA 1 and CA 2 and returns these to the initiating application
- Advantage:
  - Aggregate fabric load is reduce—fewer packets exchanged across inter-switch links and application links
  - Lower load reduces probability of congestion which improves effective collective latency
  - Significantly reduces the number of interrupts and packet processing load on compute nodes due to accelerator offload



This slide illustrates example steps to execute a broadcast collective using collective accelerators. In this example, the collective accelerators optimize communications to reduce fabric load, the potential of congestion events, and to improve application performance (e.g., eliminate software overheads, interrupts, etc. used in software-only implementations). A collective is initiated by an application or on behalf of an application using middleware software. If software has explicit knowledge of a specific collective accelerator, then it can directly communicate with it using a collective request packet. If it does not have explicit knowledge of a specific collective accelerator, then it can use multicast encapsulation (even if the switches do not support multicast packet replication) to transparently target the initial collective accelerator which takes over executing the collective across all collective accelerators. Multicast encapsulation enables a unicast packet to be encapsulated and distributed to a multicast group.



Example Collective Packet Format



- Collective Group ID—identifies the set of components participating in a collective
- Collective ID—identifies a specific collective operation
- REQCTXID / RSPCTXID—Requester and Responder contexts identifiers used to locate resources
- Data Type—size and type of data to target, e.g., 8 / 16 / 32 / 64 / 128 / 256-bit data, Integer / FP, etc.
- Payload (Data Type)
- CH—Indicates if a completion handler is to be invoked upon receiving the packet

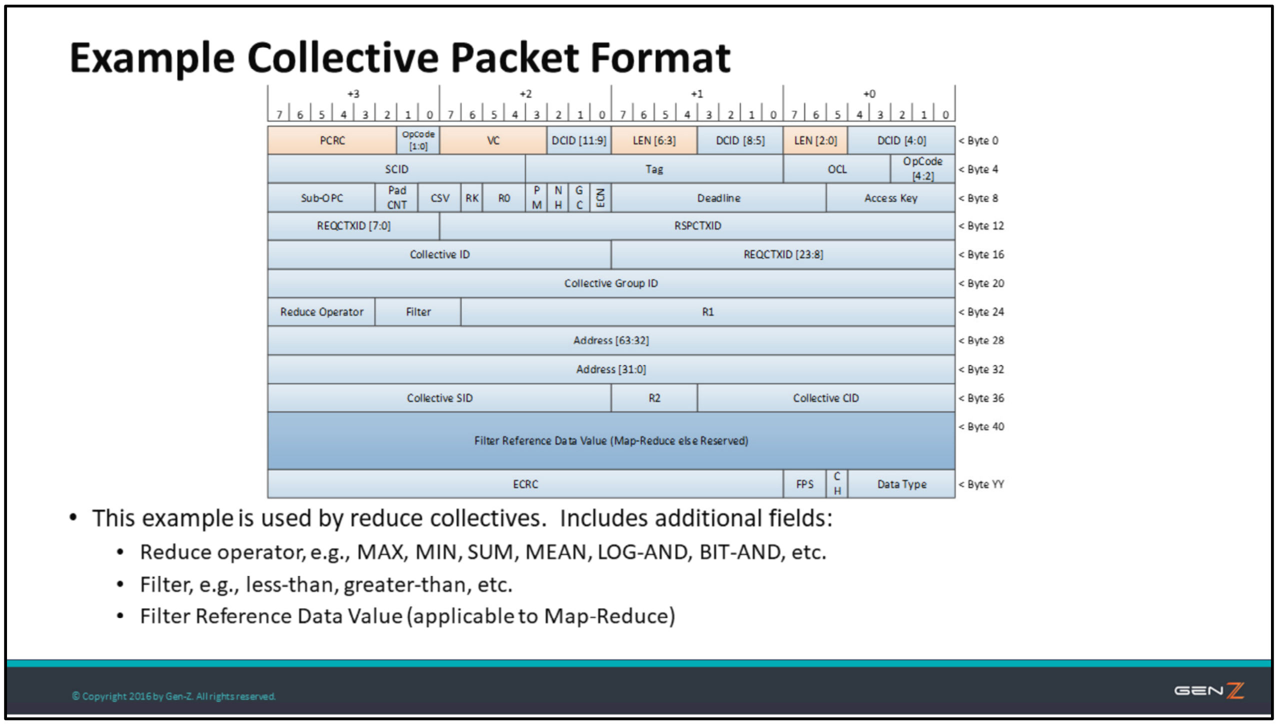
All collective packets contain a set fields used to identify the collective group and the specific collective operation within that group. This enables a Gen-Z fabric to support multiple groups and outstanding collective operations.

Collective operations use the CTXID OpClass. This OpClass includes Requester and Responder context identifiers (application handles) to quickly and easily identify the application resources (simplifies hardware implementations)

Collectives can use a variety of data type sizes, signed / unsigned integer and floating point.

Though this example contains a payload field, not all collective packet formats do.

If a completion handler needs to be invoked, then the component sets CH = 1b. This enables the Requester to dynamically signal completion handling should be invoked to inform the application / middleware of the request packet’s arrival (solutions using collective accelerators may not require such invocations depending upon the implementation).



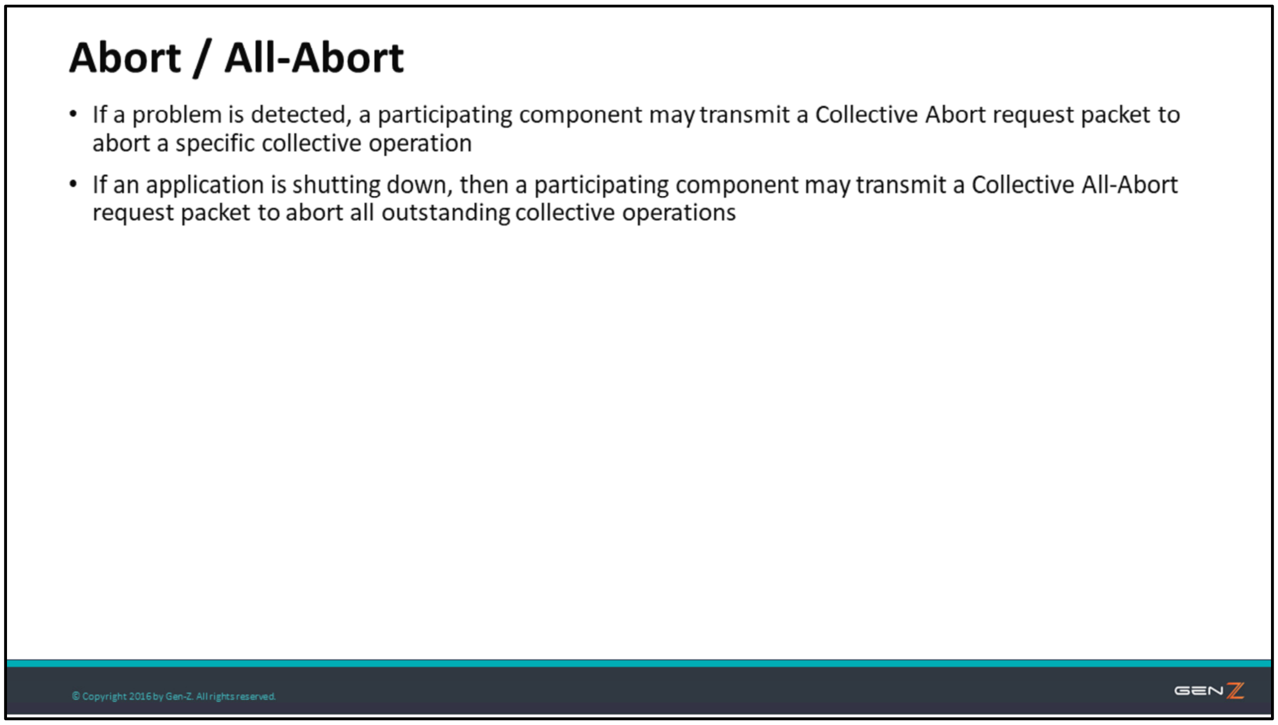
Example Collective Packet Format

PCRC	OpCode [1:0]	VC	DCID [11:9]	LEN [6:3]	DCID [8:5]	LEN [2:0]	DCID [4:0]				< Byte 0
SCID	Tag	OCL	OpCode [4:2]								< Byte 4
Sub-OPC	Pad CNT	CSV	RK	R0	PM	NH	GC	ECN	Deadline	Access Key	< Byte 8
REQCTXID [7:0]	RSPCTXID										< Byte 12
Collective ID	REQCTXID [23:8]										< Byte 16
Collective Group ID											< Byte 20
Reduce Operator	Filter	R1									< Byte 24
Address [63:32]											< Byte 28
Address [31:0]											< Byte 32
Collective SID	R2	Collective CID									< Byte 36
Filter Reference Data Value (Map-Reduce else Reserved)											< Byte 40
ECRC	FPS	CH	Data Type								< Byte YY

- This example is used by reduce collectives. Includes additional fields:
  - Reduce operator, e.g., MAX, MIN, SUM, MEAN, LOG-AND, BIT-AND, etc.
  - Filter, e.g., less-than, greater-than, etc.
  - Filter Reference Data Value (applicable to Map-Reduce)

GEN Z

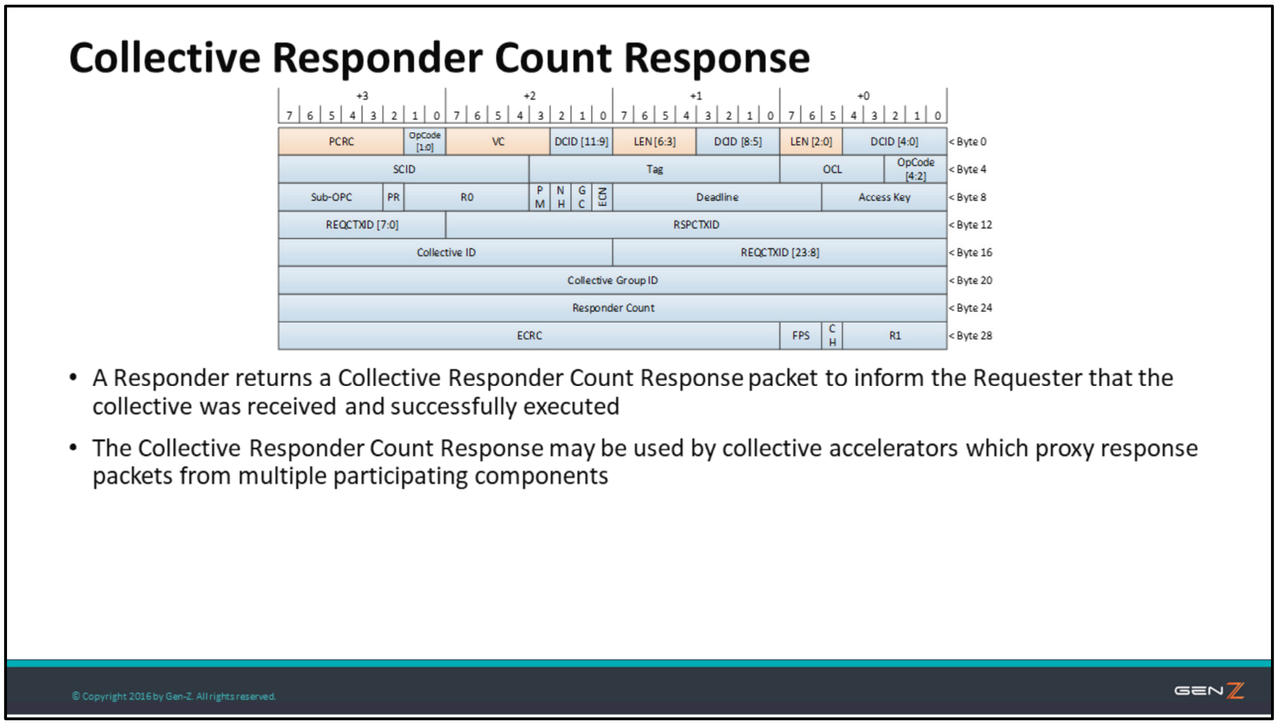
This packet format is used by reduce collectives. It contains additional fields include a Reduce Operator (what reduction to be performed), and a Filter to indicate what comparison to perform, in the case of a Map-Reduce operation, the packet includes the Filter Reference Data.



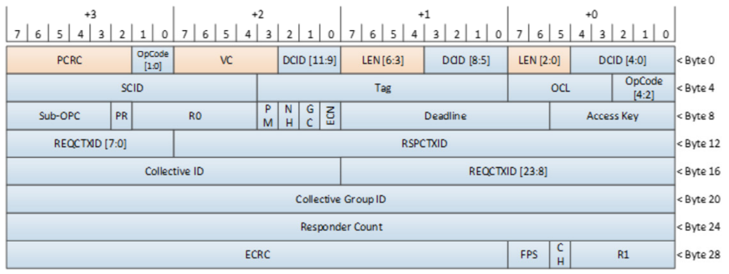
Abort / All-Abort

- If a problem is detected, a participating component may transmit a Collective Abort request packet to abort a specific collective operation
- If an application is shutting down, then a participating component may transmit a Collective All-Abort request packet to abort all outstanding collective operations

A collective Abort impacts a single collective group and a single collective operation. A collective All-Abort impacts all outstanding collective operations associated with a specific collective group.

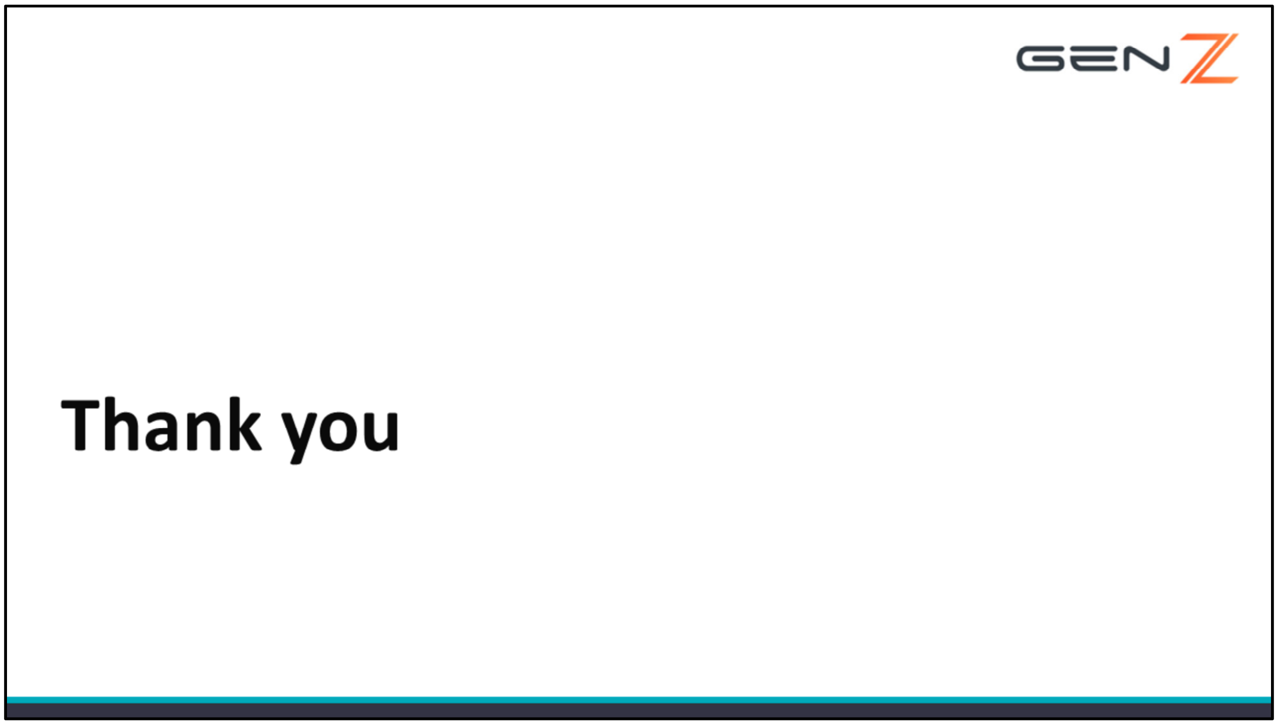


Collective Responder Count Response



- A Responder returns a Collective Responder Count Response packet to inform the Requester that the collective was received and successfully executed
- The Collective Responder Count Response may be used by collective accelerators which proxy response packets from multiple participating components

Unless an error was detected, a Collective Responder Count Response packet is returned for each collective request packet. The Collective Responder Count Response can represent one or more Responders, enabling hierarchical collective solutions and / or collective accelerators to aggregate the results, i.e., response packets, from multiple Responders.



Thank you

This concludes this presentation on collectives. Thank you.