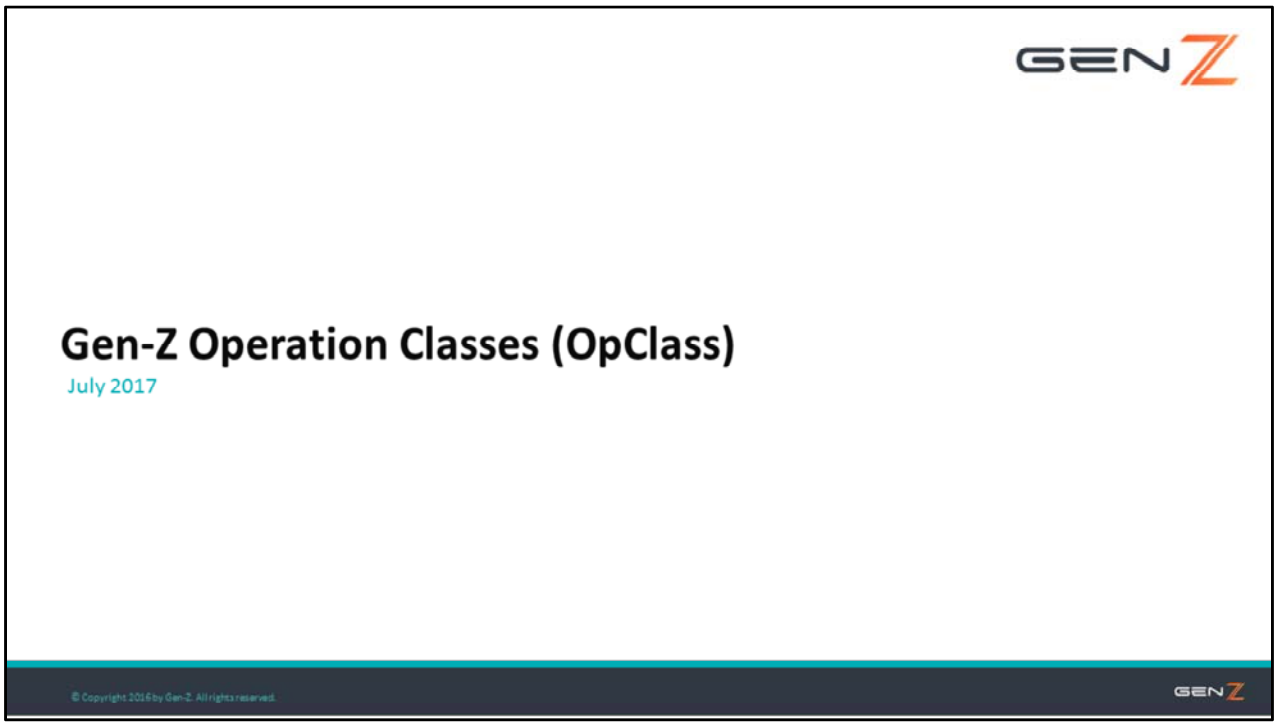


This presentation covers Gen-Z Operation Classes, also referred to as OpClasses.

Disclaimer

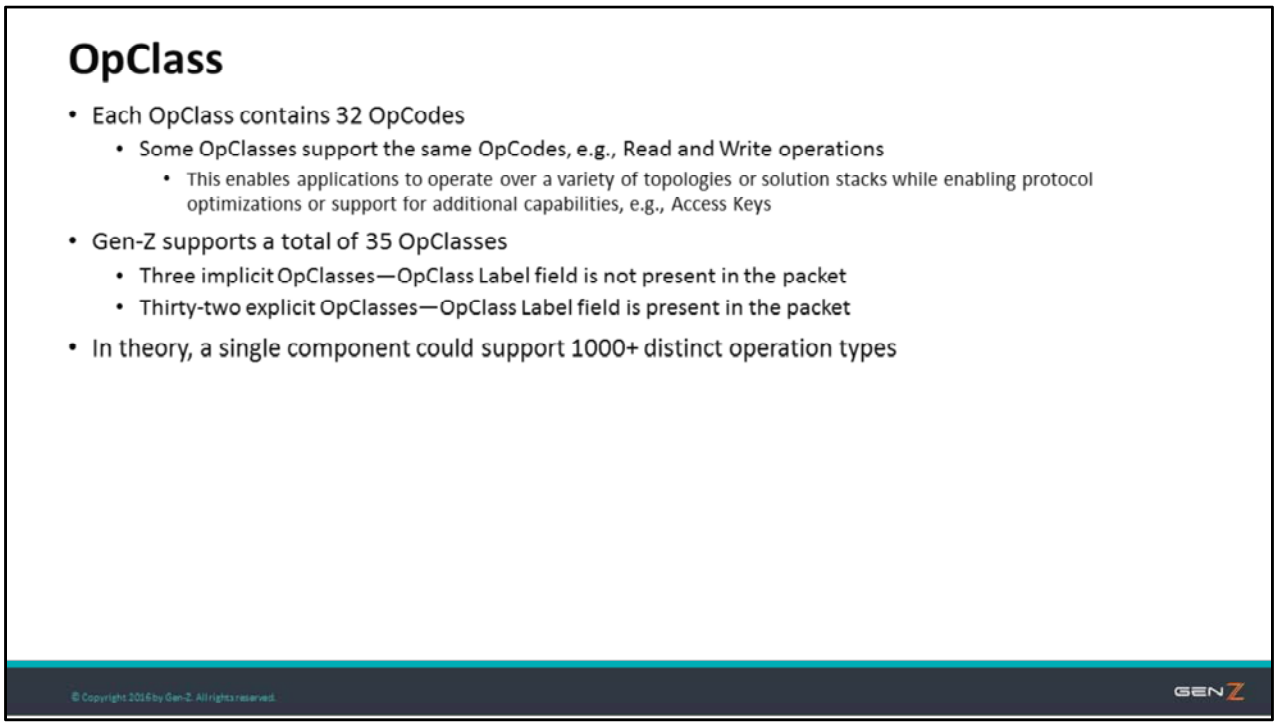
This document is provided 'as is' with no warranties whatsoever, including any warranty of merchantability, noninfringement, fitness for any particular purpose, or any warranty otherwise arising out of any proposal, specification, or sample. Gen-Z Consortium disclaims all liability for infringement of 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/



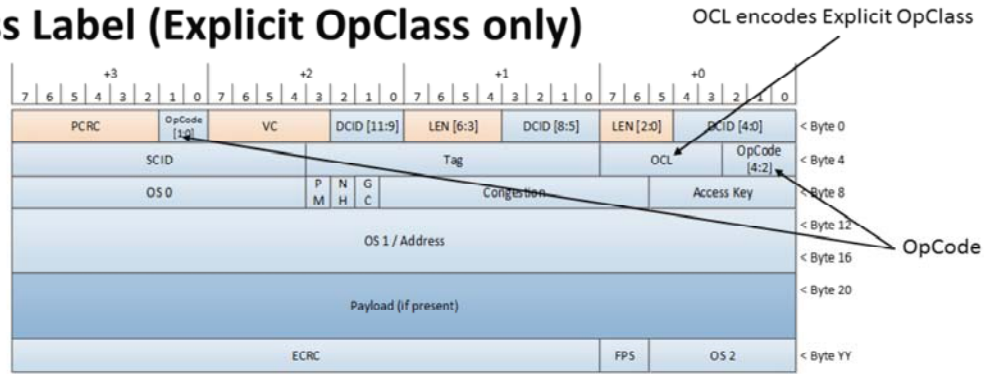
Gen-Z organizes operations into OpClasses. Each OpClass contains 32 OpCodes, where each OpCode represents a unique operation, e.g., a Read or Write operation.

Gen-Z supports a total of 35 OpClasses. Three implicit OpClasses and 32 explicit OpClasses.

Implicit OpClasses are configured on a per component interface basis and are intended for use in point-to-point, mesh, and daisy-chain topologies. Implicit OpClass packets do not contain an OpClass Label field.

Explicit OpClasses are intended for use in point-to-point and switch-based topologies. Explicit OpClass packets contain an OpClass Label field that explicitly identifies the OpClass.

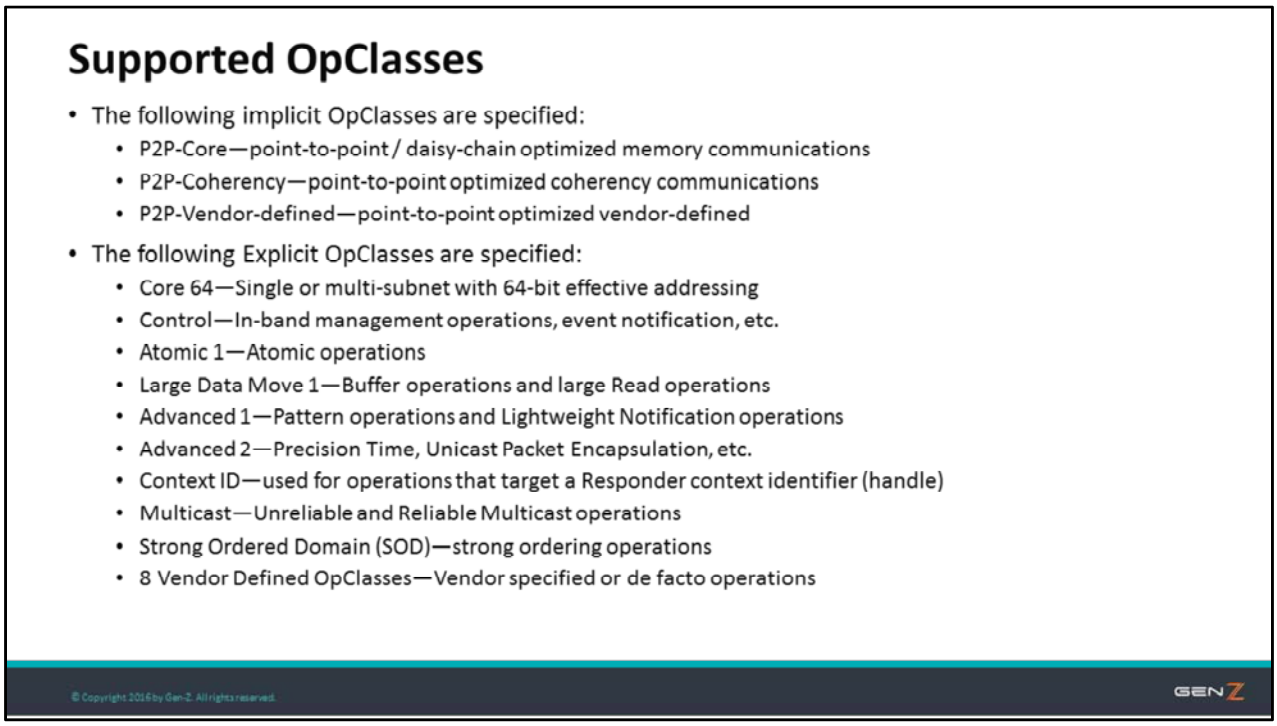
OpClass Label (Explicit OpClass only)



- The OCL field is a 5-bit field that contains the encoded OpClass identifier
- The OpCode field is a 5-bit field that indicates the operation associated with the indicated OpClass

GEN Z

This slide illustrates a generic explicit OpClass packet. There are two fields of interest: the OpCode field identifies the specific operation type and the OpClass Label (OCL) field identifies the specific OpClass. Each is a 5-bit field respectively providing 32 OpCodes and 32 Explicit OpClasses.



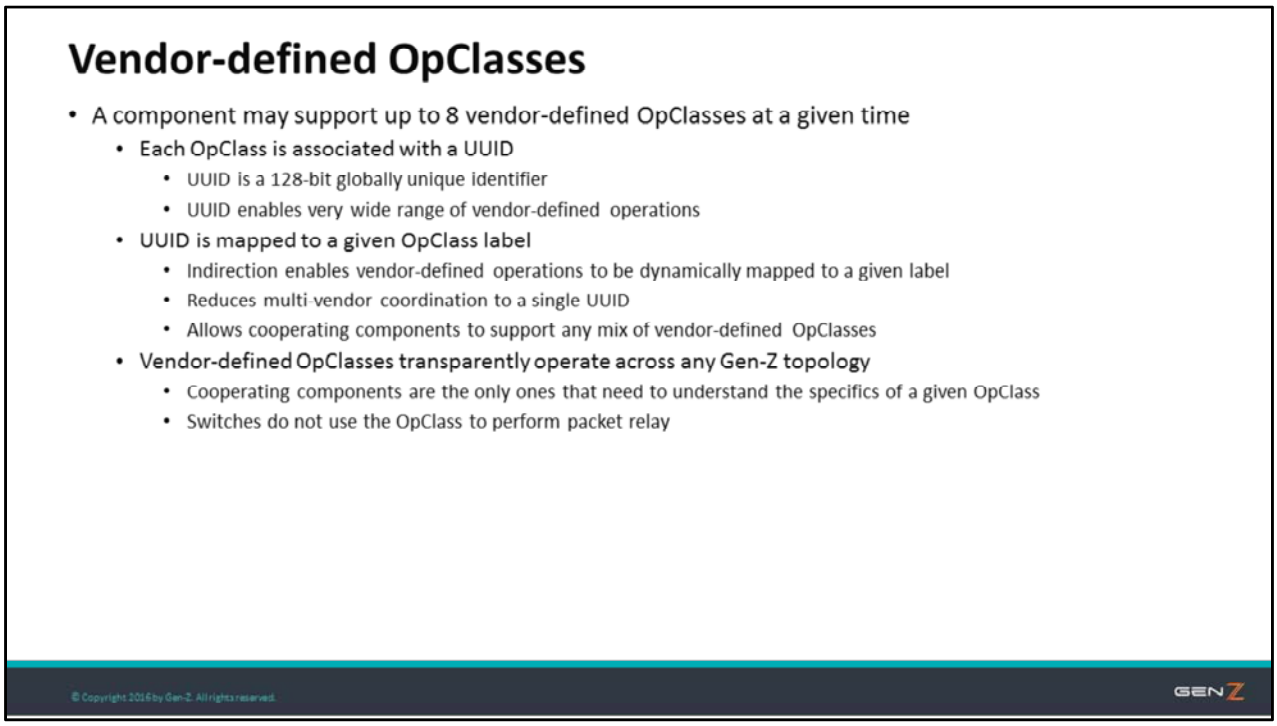
The following three implicit OpClasses are specified: P2P-Core, P2P-Coherency, and P2P-Vendor-defined.

- The P2P- Core OpClass is optimized for point-to-point and daisy-chain topologies used to connect processors with memory modules.
- The P2P-Coherency OpClass is optimized for point-to-point and mesh topologies used to connect coherent components, e.g., a processor and an accelerator.
- The P2P-Vendor-defined OpClass is optimized for point-to-point solutions using a vendor-defined protocol.

The following explicit OpClasses are specified:

- Core 64 is optimized for switch-base solutions, e.g., I/O and memory expansion within a single enclosure or rack. It supports 64-bit addressing and can be used in single and multi-subnet topologies.
- The Control OpClass is used by in-band management to access control space.
- The Atomic OpClass is used to support a robust set of atomic operations compatible with multiple processor architectures.
- The Large Data Move OpClass is used to move large quantities of data using Buffer Put and Get semantics as well as large read semantics. In general, large data move operations are executed by data mover logic instead of a processor core.
- The Advanced 1 and Advanced 2 OpClasses are used to support pattern operations, lightweight notification, precision time, unicast packet encapsulation, etc.
- The Context ID OpClass is used to target a Responder Context Identifier (handle). This enables operations to target a specific instance of a service, e.g., to transmit messages used to a specific eNIC (emulated NIC) or to communicate collective operations to a specific application instance.
- The Multicast OpClass is used to transmit unreliable and reliable multicast operations. This enables Requesters, Responders, and switches to easily delineate unicast from multicast operations, simplifying hardware designs.
- The Strong Ordered Domain (aka SOD) is used to exchange strongly ordered operations, i.e., these packets use sequence number based to enable a Responder to detect missing or out-of-order packets.

- Vendor-defined operations will be covered in the next slide.

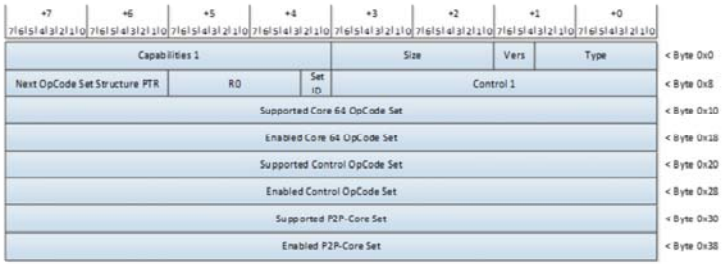


In addition to these standard OpClasses, Gen-Z supports 8 vendor-defined OpClasses. These OpClass may be used to communicate vendor-specific operations or de facto standard operations. A UUID is used to uniquely identify each OpClass. This enables the architecture to support nearly any number of vendor-defined operations and, by using UUIDs, to significantly simplify management and enable vendors to innovate and collaborate without requiring a central management authority.

The UUID is not present on the wire since that would be extremely inefficient. Instead, it is mapped to an OpClass Label field. Hence, any component can support up to 8 different vendor-defined OpClasses at a given time, and these are transparently transported across the Gen-Z topology.

OpCode Set Structure

- OpClasses are managed via the OpCode Set structure
- For each OpClass, there are two fields:
  - Supported OpCode Set
  - Enabled OpCode Set
- Each field is a set of 32 2-bit sub-fields
  - If sub-field == 0x0, then this indicates an unsupported or non-enabled operation
  - If sub-field == 0x1, then this indicates a supported or enabled operation as a Requester-only
  - If sub-field == 0x2, then this indicates a supported or enabled operation as a Responder-only
  - If sub-field == 0x3, then this indicates a supported or enabled operation as a Requester-Responder
- A component may support multiple OpCode Set structures to enable interoperability with a variety of disparate components or across multiple technology generations

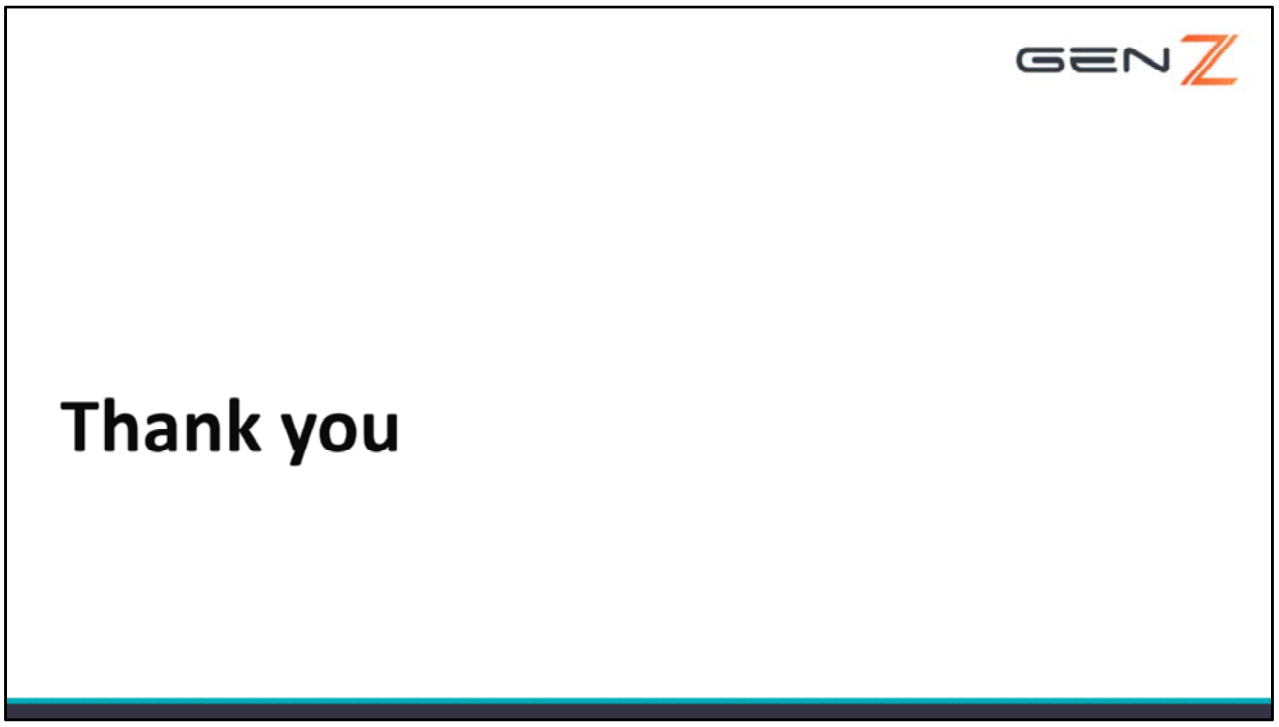


Initial fields within the OpCode Set Structure

GEN Z

OpClasses are managed through the OpCode Set structure. Two fields are provisioned per OpClass: the supported OpCode set and the enabled OpCode Set. As the names imply, the Supported field indicates which OpCodes are supported and the Enabled field indicates which OpCodes a component may use. Further, each field is composed of sub-fields to indicate each OpCode's role, i.e., whether it supports and is enabled to use an operation as a Requester, as a Responder, or as a Requester-Responder.

A component may support multiple OpCode Set structures to enable interoperability with a variety of disparate component or across multiple technology generations.



This concludes this presentation. Thank you.