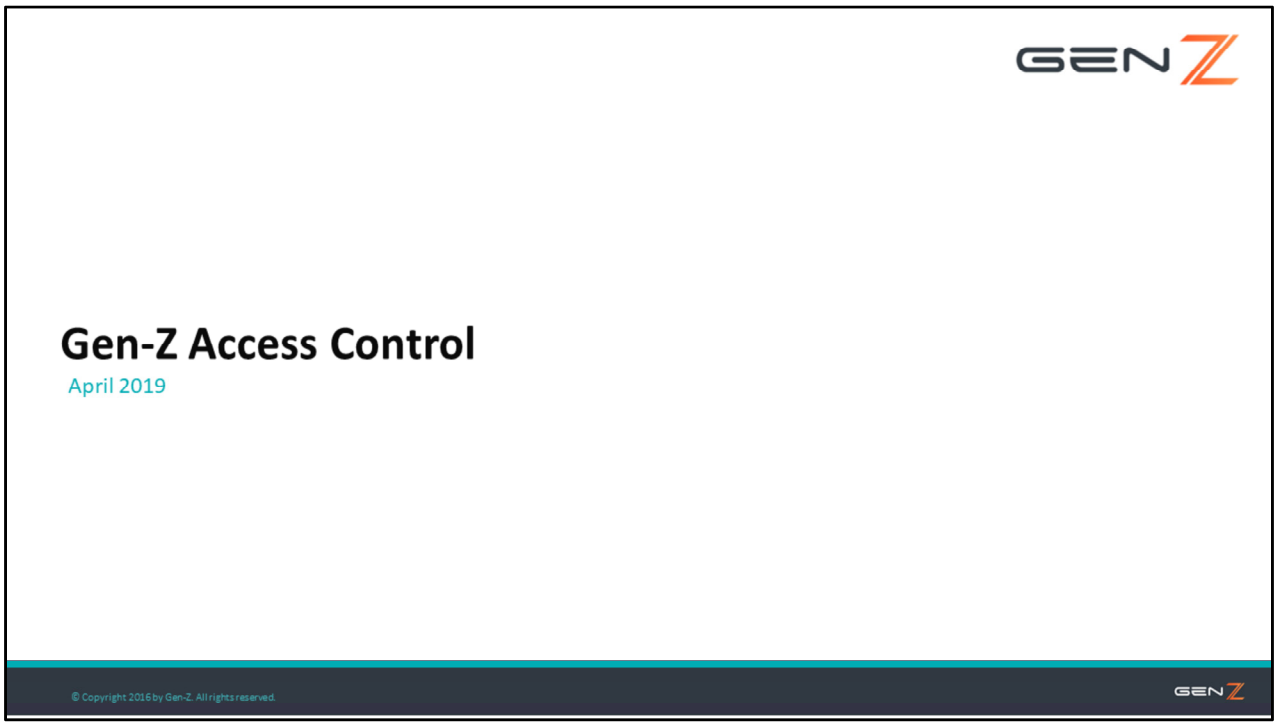


This presentation cover Gen-Z Access Control.

Disclaimer

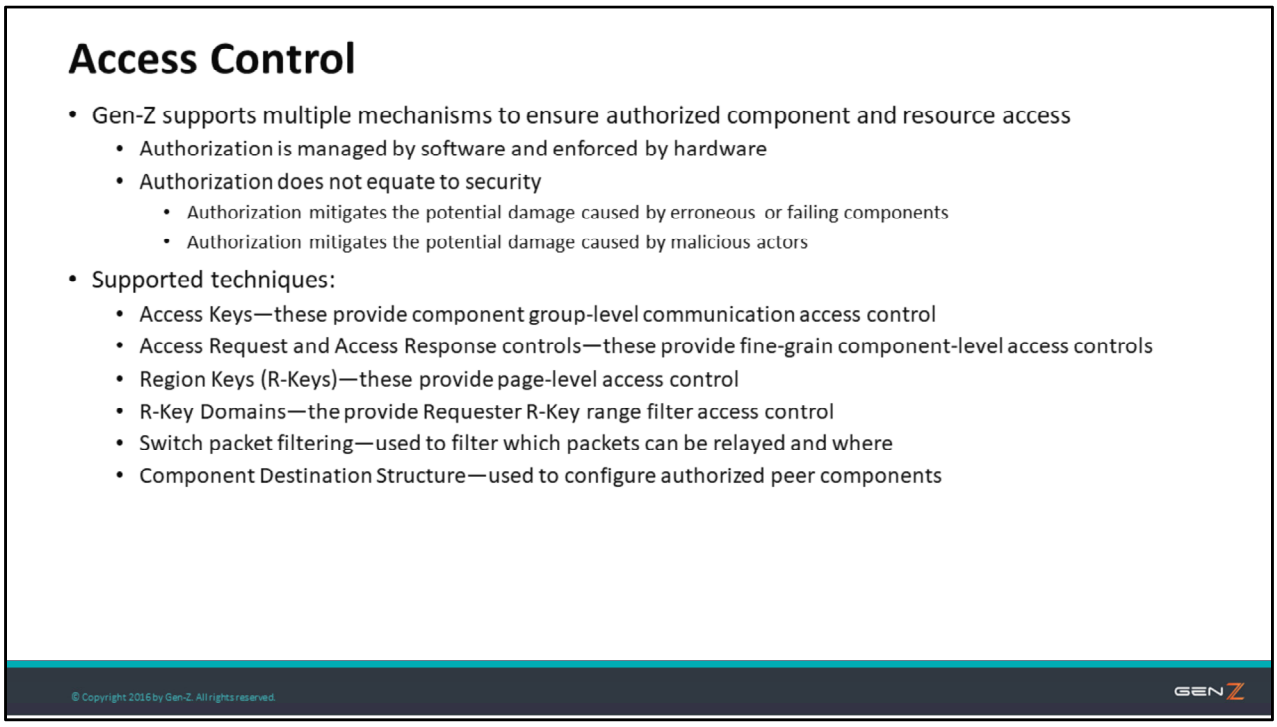
This document is provided 'as is' with no warranties whatsoever, including any warranty of merchantability, noninfringement, fitness for any particular purpose, or any warranty otherwise arising out of any proposal, specification, or sample. Gen-Z Consortium disclaims all liability for infringement of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

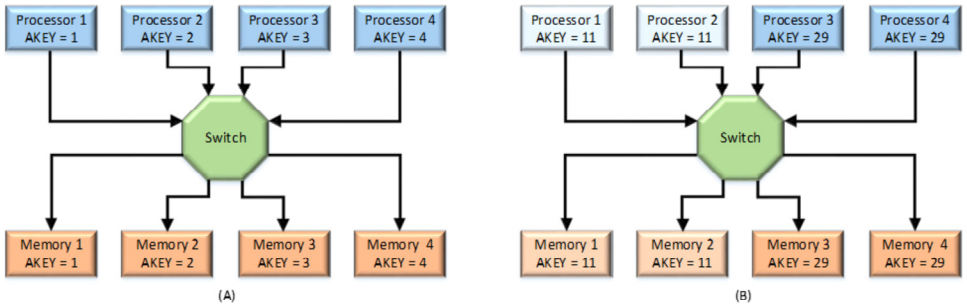


Gen-Z supports multiple mechanisms to enforce access control and access permission validation in hardware with minimum resources and performance impact (most validation can occur in parallel with other aspects of packet validation, thus eliminating any latency impacts).

Access control is used to determine whether a component is authorized to communicate with another component and, if supported, if the component is permitted to access a given resource. Authorization is determined entirely in software (application / middleware / management), and is enforced in hardware (simple, high-performance). Access control is not a substitution for a robust security solution. Access control is sufficient to mitigate potential damage caused by erroneous or failing components or to quickly isolate malicious actors, but Access Control does not prevent packet spoofing or anti-replay attacks (see Gen-Z Security for details on how these attacks are handled).

Gen-Z supports multiple techniques to enable components to incorporate access control as required by specific solution stacks.

Access Keys



- Access Key is an opaque identifier used to enforce component isolation
- Access Keys may be used in any topology and with any mix of component types
- Access Keys may span multiple subnets
- Access Key space is 2^6
- Default Access Key is 0x0

GEN Z

All explicit OpClass packets contain a 6-bit Access Key field. Though small, this field is sufficient to meet nearly market needs. Being small also enables Access Keys to be easily implemented on any component interface with no latency impacts, broadening the appeal and potential ubiquitous adoption and deployment within the industry. In contrast to alternatives, Gen-Z does not require 2x more keys to support full and limited access (Access Request and Access Response controls provide a simpler, more efficient and robust alternative to using Access Keys for such purposes), hence, Gen-Z supports the same number of keys as are generally implemented in alternatives.

Access Keys may be used in any topology, and can span multiple subnets. For very large scale-out topologies, management can use a combination of packet relay tables and Access Keys to ensure components can only communicate with authorized peers, or through authorized Requester, Responder, TR, and switch interfaces. Further, by using packet relay tables and interface-level Access Key validation, larger topologies can be partitioned such that the effective number of Access Keys is multiplied, eliminating the need for a larger number.

In example A, eight components share a common switch. Each processor-memory component pair is configured with a unique Access Key. The switch interfaces and the component interfaces can enforce Access Key validation to prevent component pairs from

accessing another pair’s resources.

Example B illustrates Access Keys applied to larger component groups, e.g., a rack scale or multi-rack solution.

Access Keys (continued)

- Explicit OpClass packets contain an Access Key field
  - Access Key field may contain any Access Key value
- Point-to-point optimized OpClass packets do not contain an Access Key field
- Requester and Responders select and validate Access Keys at the component level
  - Enables any egress interface able to reach the destination to be used to transmit packets.
  - Validate that the correct Access Key is used to communicate with a peer component
- Requesters, Responders, and switches validate Access Key at ingress and egress interfaces
  - Uses a bit mask where each Bit_K corresponds to Access Key K. If Bit_K == 1b then the packet is permitted to be relayed or transmitted to the next component
- Access Keys are managed through the:
  - Component PA structure
  - Interface structure

GEN Z

Requesters and Responders enforce Access Keys at the component level. This enables the Access Key to be determined early in the packet creation time which can simplify implementation and eliminate any latency impacts.

A Requester, Responder, or switch interface validates the Access Key field upon packet receipt and prior to packet transmission (this can be done in parallel with other packet processing which eliminates any latency impacts). Interface-level Access Key validation prevents a component from processing or transmitting unauthorized packets.

Access Key validation is performed using a simple bit mask where every bit indicates if the corresponding Access Key is permitted or not.

Access Keys are managed at the component level through the Component PA (peer attribute) structure, and on a per component interface level through the Interface structure. The Component PA structure contains a 6-bit Access Key value per peer table entry (to reduce implementation cost / complexity, these entries can be shared by multiple peers). The Interface structure uses two 64 bit masks.

Access Request / Access Response

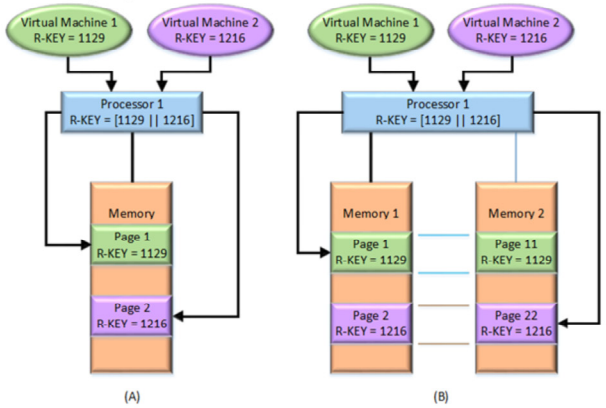
- The Component PA structure contains a set of tables used by Requesters and Responders to support wildcard or fine-grain peer access controls
  - For example, a Responder memory component might support only a single Access Key or set of peer attributes to communicate with all authorized Requesters.
  - A Requester might communicate with multiple Responders, each requiring a different Access Key or a unique set of peer attributes.
- Whether a wildcard version or a specific table entry, each Peer Attribute field contains a two-bit sub-field that indicates what type of access is permitted:
  - No Access is permitted
  - Access is permitted, however an R-Key is required on all applicable packets
  - Full Access / Trusted Component—R-Keys can be ignored / not used

GEN Z

The Peer Attribute field contains a set of bits that describe the type of communications permitted or required when communicating with one or more peer components. Whether using a wildcard set of values or fine-grain table entries, management configures the specific behaviors.

The two-bit Access Control sub-field operates independent of the Access Key. This eliminates the need for an Access Key to grant all-or-nothing access to a peer component.

Region Keys (R-Keys)



- A R-Key is an opaque identifier used to enforce page read and write access within a component
- R-Keys may be used by any component type
- R-Keys may be used to protect Data Space and Control Space
- R-Key space is 2^{32}
- Default R-Key is 0x0

GEN Z

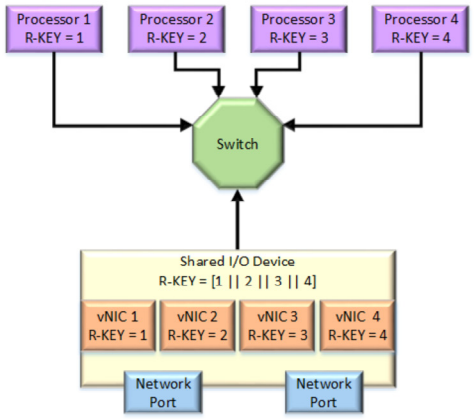
An R-Key is a 32-bit opaque identifier used to validate access permission to an addressable resource at page-level granularity. R-Keys are managed by application or middleware software.

R-Keys enable multiple Requesters to simultaneously share a Responder without exposing all addressable resources to all Requesters. Example A illustrates two virtual machines operating on a single Requester to simultaneously access a memory module, though each can access only a subset of pages with matching R-Keys. Example B illustrates the same virtual machines accessing memory regions that span multiple memory modules (e.g., discrete or interleaved memory). Each can access only its corresponding pages.

In general, a Requester will use a ZMMU (Gen-Z memory management unit) to transparently access Gen-Z addressable resources. Each page table entry (PTE) contains the read-only or the read-write R-Key as well as other Responder-specific information (e.g., component identifiers).

If a page is not protected by an R-Key, then it will be configured to use the Default R-Key. If a Default R-Key is detected within the Requester PTE entry, then the R-Key field need not be present in the packet (this improves protocol efficiency).

R-Keys (continued)



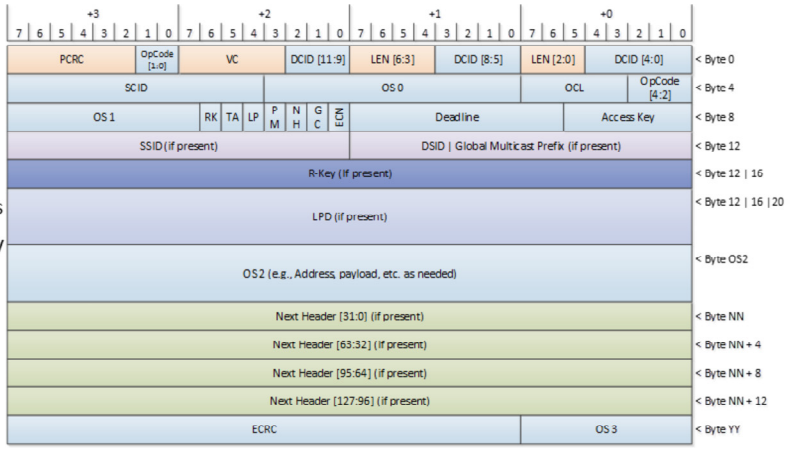
- R-Keys can be used to isolate resources to a particular processor, virtual machine, application, etc.
- R-Keys can span multiple memory pages

GEN Z

In this example, multiple processors are simultaneously sharing an I/O device (e.g., a Gen-Z Logical PCI Device (LPD)). Each Requester application uses a distinct R-Key to access the corresponding virtual NIC resources. Components that take advantage of Gen-Z's architecture, e.g., multipath, R-Keys, etc. can safely share a Responder without fear that their resources will be compromised by a peer component's access.

R-Keys (continued)

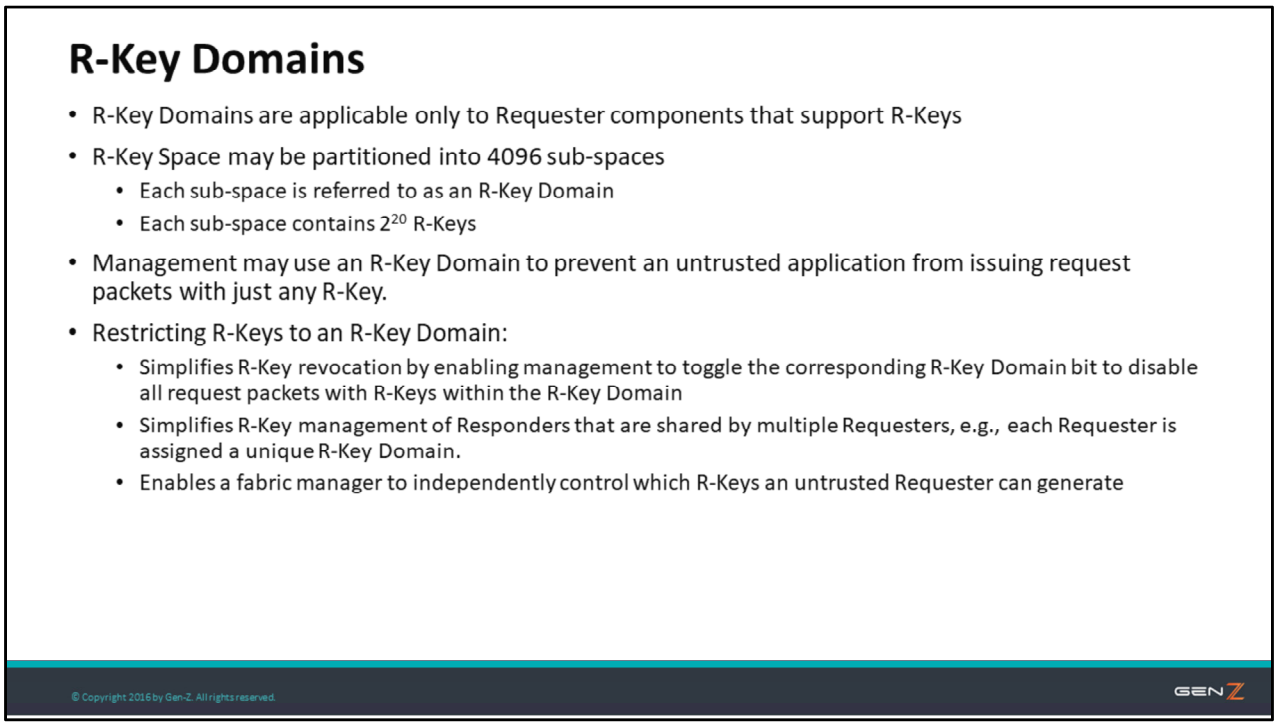
- Only explicit OpClass request packets that contain an RK bit may contain an R-Key field
  - If RK == 1b, then the R-Key field is present
- R-Keys authorized two types of access permission
  - RO R-Key is used to provide read-only access
  - RW R-Key is used to provide read-write access
- Responders authorize read-write or read-only access based on which R-Key is provided to a Requester
- R-Keys may be configured through a Requester or Responder ZMMU (if supported)



The R-Key field can be dynamically added to any explicit OpClass request packet that contains the RK bit field.

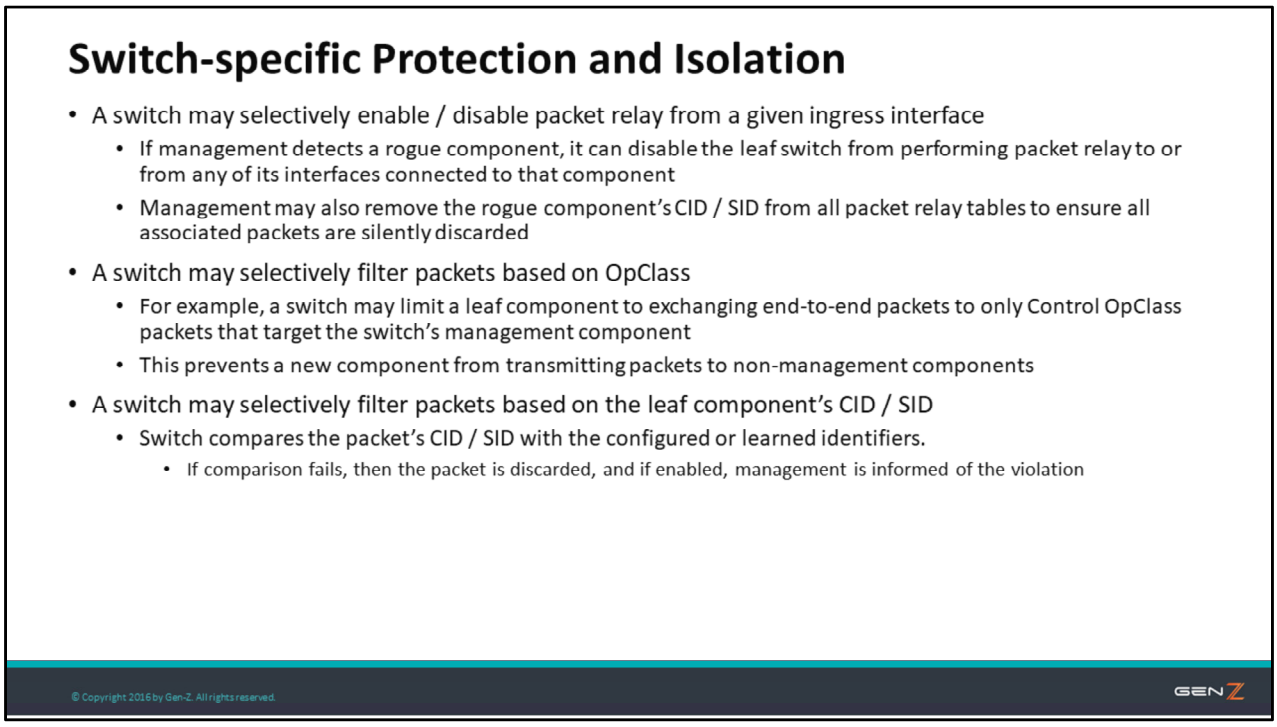
R-Keys can be used to simplify multiple-reader, single-writer applications. To support multiple-reader, single writer applications, R-Keys can be configured to enable multiple Requesters to simultaneously read shared memory using the read-only R-Key, and configured to enable a single Requester to modify shared memory using the read-write R-Key.

A Requester transparently accesses the R-Key associated with a given Responder resource through the Requester ZMMU. A Responder uses a Responder ZMMU to configure read-only and read-write R-Keys for each page(s).



To prevent a Requester from transmitting just any R-Key, a component uses R-Key Domains (RKDs) to validate ranges of R-Keys that a Requester is permitted to use in request packets. The R-Key Space is partitioned into 4096 sub-spaces (upper 12 bits). Hardware uses a bit-mask to indicate whether a given RKD is permitted or not by using the upper 12 bits as an index into the bit mask (if the bit == 1b, then the R-Key is permitted). Since RKD resource requirements are small and validation can be done in parallel with other aspects of request packet creation (no latency penalty), RKD functionality is mandatory in Requesters that support R-Keys.

RKDs have numerous advantages that stem from their simplicity to implement and manage.



To minimize latency, Gen-Z switches maintain packet relay tables on a per interface basis. This has the side benefit of enabling each component interface to act as a first-pass filter by validating whether a given destination component may be reached through this interface. If a rogue (erroneous or malicious) component is detected, the switch discards the packets and informs management of the violation. Management can take additional actions to automatically disable the rogue component (e.g.,, reset or power down the component) or to isolate the component from transmitting packets through any other switch.

A switch may perform OpClass packet filtering. For example, when a component is first discovered, a switch can be configured to silently discard all non-Control OpClass packets from being transmitted by the component, i.e., limits the component to only respond to in-band management requests and take no other actions.

Through management configuration or link-level discovery, a switch can filter packets based on the leaf component's identifiers. If a component attempts to transmit a packet masquerading as a different component, then the switch discards the packets and informs management of the violation.

Access Error Violation Handling

- The detecting component discards the packet
  - If a Responder and the request packet uses Reliable Delivery, then generate a corresponding response with the Reason field set to indicate AE
- If the detecting component supports the Component Error and Signal Event structure, then the configured behavior is taken:
  - If AE Detect is enabled, then update error status
  - If Trigger Containment is enabled, then trigger component error containment
  - If Signal Error is enabled, then inform management of the AE violation
    - If configured, generate component local interrupts to inform up to two local management entities, e.g., firmware and an OS
    - If configured, generate an Unsolicited Event packet to inform a fabric-attached management component

GEN Z

Gen-Z architecture enables management to take a variety of actions upon detecting errors or access violations. Further, it can inform multiple management entities of different or the same event, e.g., within a single enclosure, it can inform system firmware and an operating system or hypervisor. If a multi-enclosure / rack-scale solution, it can inform a fabric manager. In all cases, it can inform the application through acknowledgment packets that indicate the detected highest-precedence error.

Thank you

This concludes this presentation. Thank you.