GEN-Z CONSORTIUM™

Gen-Z Fabric Management Specification

Version 1.0

Change Summary

1.0 Initial Release

Contents

Figures

Tables

1. Introduction

1.1. Reference Documents

Gen-Z Core Specification 1.1, https://genz.causewaynow.com

Gen-Z Physical Layer Specification, https://genz.causewaynow.com

Gen-Z Mechanical Form Factor Specification, https://genz.causewaynow.com

PCI Express Base Specification, version 4.0. http://www.pcisig.com

PCI Firmware Specification, version 3.2, http://www.pcisig.com

UEFI Specification, version 2.6, http://uefi.org

ACPI Specification, version 6.1, http://uefi.org

Redfish Specification 1.9.0, https://redfish.dmtf.org

1.2. Documentation Conventions

Shall, Should, May, and Can

This specification adheres to Section 13.1 of the IEEE Specifications Style Manual, which dictates use of the words 'shall', 'should', 'may', and 'can' in the development of documentation, as follows:

- The word *shall* is used to indicate mandatory requirements strictly to be followed in order to conform to the Specification and from which no deviation is permitted (*shall* equals *is required to*).
- The use of the word *must* is deprecated and shall not be used when stating mandatory requirements; *must* is used only to describe unavoidable situations.
- The use of the word *will* is deprecated and shall not be used when stating mandatory requirements; *will* is only used in statements of fact.
- The word *should* is used to indicate that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain course of action is deprecated but not prohibited (*should* equals *is recommended that*).
- The word *may* is used to indicate a course of action permissible within the limits of the Specification (*may* equals *is permitted*).
- The word *can* is used for statements of possibility and capability, whether material, physical, or causal (*can* equals *is able to*).

Normative vs. Informative

All sections are normative, unless they are explicitly indicated to be informative.

Capitalization

Some terms are capitalized to distinguish their definition in the context of this document from their common English meaning. Words not capitalized and not defined within this document's Glossary have their common English meaning.

Bit fields are capitalized to improve legibility.

The first letter of a term composed of multiple words concatenated as follows (A_B_C) is capitalized.

Operation Code and Operation Class names are capitalized.

Numbers and Number Bases

Unless explicitly stated otherwise by this specification, numerical values without qualifiers are decimal. This specification uses the following qualifiers:

- Hexadecimal numbers are written with a '0x' prefix followed by a mix of digits 0 through 9 and / or upper case English letters A through F, e.g., 0xFFFF or 0x80. Hexadecimal numbers larger than four digits are represented with a space dividing each group of four digits, as in 0x1E FFFF EFFF.
- Binary numbers are written with a lower case 'b' suffix, e.g., 1001b and 10b. Binary numbers larger than four digits are written with a space dividing each group of four digits, as in 1000 0101 0010b.

A dash between two numbers represents a range of values, e.g., 0-7 represents zero to seven inclusive.

A colon between two numbers represents a range of bits, e.g., Bits 7:0 represents a binary value of bits seven to zero inclusive.

Standard Units of Time

Unit	Description
ms	Millisecond—10^{-3} seconds
µs	Microsecond—10^{-6} seconds
ns	Nanosecond—10^{-9} seconds
ps	Picosecond—10^{-12} seconds
fs	Femtosecond—10^{-15} seconds

Reserved

The following applies to the term 'Reserved':

- The contents, state, or information are not specified at this time.
- Any field, feature, capability, etc. marked 'Reserved' is subject to change.

- Components shall not use any field, feature, capability, etc. that is marked 'Reserved'.
- All Reserved control structure fields shall be read-only, and shall return 0x0 when read (single or multi-bit fields).
- Reserved encodings of any control structure, packet fields, etc. shall not be used.
- Any implementation dependence on a 'Reserved' field value, or encoding will result in a non-compliant implementation. The functionality of such an implementation cannot be guaranteed in this or any future revision of this specification.
- A Reserved protocol field shall be transmitted as zero and ignored upon receipt.

Developer Notes

Developer Notes do not specify normative or optional requirements. They are included for clarification and illustration only. These notes are delineated by: ***Developer Note:*** *Text*

State Machine Conventions

State machines describe the behavior of a component, an interface, a link, or the physical layer. A state machine does not imply the internal design or implementation. State machines use the following conventions:

- Each state machine is represented by a rectangular box.
  - The top section of the box contains the state name.
  - The bottom section of the box contains the actions which occur in the state.
- Transition arrows indicate state transitions which are made when the expression next to the arrow is satisfied.
- A transition arrow which does not originate in a state indicates a global transition. Such a transition will occur regardless of the current state.
- If no exit condition for state is met, the state machine remains in the current state.
- A logical "OR" is represented by a "+".
- A logical "AND" is represented by a "*".
- If a conflict occurs, state machine figures take precedence over descriptive text.

Flow Chart Conventions

Flow charts illustrate packet processing, validation, etc. Flow charts use the following conventions:

- A 'Go to' represents a jump to another flow chart to complete processing.
- A 'Perform XXX ( )' represents a branch to a flow chart that performs additional processing before returning to continue processing in the current flow chart. Conceptually, this is similar to invoking a sub-function.

Table Entry Conventions

Unless explicitly stated within this specification, common table entries are defined as follows.

Table Entry	Description

M	Mandatory—the field or associated semantic is treated as a 'shall'.
O	Optional—the field or associated semantic is treated as a 'shall if implemented'.

1.3. Scope of Document

This document IS / DOES

- Deal primarily with in-band management of switched, composable Gen-Z fabrics
- Establish a general view of a Gen-Z fabric's management stack
- Define key system features and requirements which must be supported by the Gen-Z fabric's management stack
- Delineate between 'fabric' management of basic Gen-Z fabric switch and endpoint components versus the 'resource' management of Gen-Z visible resources accessible through these components
- Define the manager sub-types associated with 'fabric' management
- Define the roles of the sub-types of 'fabric' management
- Define a hierarchy of the various Fabric Managers and Resource Managers
- State key axioms assumed which enable plug and play managers and components compatible with them
- Define the general policies and work flows used by the 'fabric' managers to execute their roles
- Define the *required and recommended* control space features utilized by Fabric Managers to execute their roles
- Define a general fabric initialization and integration strategy compatible with topologies containing any number of Fabric Managers
- Define a communication strategy between various Fabric Managers
- Define the nature of interfaces and APIs between 'Fabric Managers' and 'resource managers'
- Define the nature of interfaces and APIs between 'Fabric Managers', 'resource managers', and a compute node's BIOS/SFW, OS, and application layers
- Call out the standards bodies responsible for defining the actual interfaces and APIs that will enable 'plug and play' behavior among multiple Fabric Managers, multiple Resource Managers, and multiple Gen-Z networks and subnets.

This document IS-NOT / DOES-NOT

- Deal with Point to Point (P2P) Gen-Z topologies and management topics specific to them
- Deal with out-of-band management solutions
- Define policies or work flows for Resource Managers
- Define where (in which node) any given manager thread executes
- Define at which privilege level any given manager thread executes (BIOS, OS, user)
- Define which manager roles are combined and which are separate threads
- Define actual API structures for the proposed interfaces

1.4. Glossary

Term	Definition
Access Key (AKEY)	An identifier within the Gen-Z protocol header used to isolate packet exchanges to only those components that share a common Access Key.
BIOS	Basic Input/Output System. This is the Windows' platform term for the initial System Firmware program that is started when the system CPU is released from reset. In this document, BIOS and System Firmware are used interchangeably.
Bridge (host bridge)	A component which translates host interface I/O transactions or memory read and write transactions directly into appropriate Gen-Z packets and vice versa.
Boundary Link	Any link that connects two Gen-Z components that are not in the same manager domain, that are not in the same CID namespace, or where a component on one end of the link is un-managed, un-powered, in-operative, or not present. See *Section 4.4 Securing Boundary Links* for details.
C-CFG	This state is used to configure the component when using in-band configuration. If a component supports only *Out-of-band Management*, then it does not enter this state. See Gen-Z Core Specification for details.
CID (Component Identifier)	An ID value that uniquely identifies a component within a single subnet.
CID Namespace	The collection of all possible CID/SID values, and whether they are currently assigned or not, that can be physically connected to a fabric.
Component PA Structure	Component Peer Attribute Structure: Used to configure the following tables used to generate and validate the peer component-specific or multicast group-specific portions of request and response packets. Includes Peer Attr, ACREG, A-Key and other fields. See Gen-Z Core Specification for details.
Control Write MSG	A Gen-Z control space packet defined by a specific Control OpClass / OpCode in the protocol header. Used by components that support *In-band Management* to exchange messages (e.g., a message between a Fabric Manager and a

	resource manager, or a message between two Fabric Managers). See Gen-Z Core Specification for details about the Control Write MSG formats.
C-UUID (Component Universally Unique Identifier)	Used to delineate component types, e.g., two memory modules from one another.
C-Up	Component Up state designation. This state is the normal operation state. A component enters this state once configuration is successfully completed. See Gen-Z Core Specification for details.
Fabric Director	A manager service that creates hierarchical management zones and is the coordinator of all the Fabric Managers in every zone.
Fabric-UUID	The Fabric-UUID is a software created 128 bit UUID that is attached to the CID namespace (and thus the fabric) by the keeper of the Grand Plan and CID Namespace.
Fabric Manager	A manager service that has direct load/store access to the Gen-Z components' architected Core Control structures. For redundancy, there can be both Primary and Secondary Fabric Managers. Different components may have different Primary and/or Secondary Fabric Managers. A single component has only one Primary Fabric Manager and one Secondary Fabric Manager.
FAM	Fabric Attached Memory: persistent or volatile memory that can be accessed from a Requestor on the Gen-Z fabric
FRU	Field Replaceable Unit
FRU-UUID	Used to identify a collection of components that are treated as a single field replaceable unit (FRU) for hardware management purposes, e.g., enclosure or a mechanical form factor that contains multiple components. See Gen-Z Core Specification for details.
Grand Plan	A specific list or an assumed default list of Gen-Z inventory and administrator choices which define an overall Gen-Z management scheme, define the policies to create a complete Gen-Z CID namespace and define the policies to handle dynamic configuration changes. See Section 2.3.2 Domains, Namespaces, and the Grand Plan of this document for details.

GCID (Global Component Identifier)	A value that uniquely identifies a component within a multi-subnet topology. A GCID is the combination of a component's Component ID (CID) and its Subnet ID (SID).
HW	Hardware: physical components.
HW Prep	Low level, out-of-band setup operations required by the platform to prepare components for being fully configured by an in-band Gen-Z manager.
In-band Management	Gen-Z management commands encoded into Gen-Z Control OpClass packets and delivered to the target via internal or external Gen-Z links
Limited Scope Managers	Processes with access rights to select R-Key protected Gen-Z control structures, but not the full permissions granted to the Primary or Fabric Managers. See section 8.13 Control Structure Organization as well as section 8.35 Component C-Access Structure of the Core Specification.
Link RFC (Ready for Configuration) Packet	Transmission of a Link RFC packet indicates the component has reached an internal operational state where it can execute *In-band Management* packets, or the component is performing self-initialization that will be completed in the indicated time.
LPRT	Linear Packet Relay Table. Each ingress port associated with a Gen-Z switch structure has an LPRT which contains the allowed egress ports to which a single-subnet packet may be relayed. See Gen-Z Core Specification for details.
L-UP	Link Up state designation. Normal operation state of a link used to exchange link-local and end-to-end packets. See Gen-Z Core Specification for details.
Management Zones	Named logical collection of Fabric Manager Domains.
Manager Domain	Components managed by a single Gen-Z manager.
MGR-UUID	A UUID used to uniquely identify the logical, software manager instance. Control OpClass request packets contain both the hardware SCID (Source CID) as well as this 'software'

	MGR-UUID to completely specify the manager hardware / software stack which issued it.
MMIO	Memory-Mapped I/O Space
MSDT (Multi-Subnet Destination Table)	Used by Requesters that support multi-subnet communications to look up the allowed egress ports to which a new request may be sent. See Gen-Z Core Specification for details.
Named Specialty Managers	Processes associated with component CIDs that match named manager ID fields in the Core Control structure. These named managers share access with the Primary and Fabric Managers over specific features or functions and some related control space structures.
Out-of-Band Management	Gen-Z control structure changes affected by issuing commands to a component using means other than a Gen-Z packet communicated on a Gen-Z link. For example, simple register reads and writes signaled on an I^2C port.
PA	Physical Address: A real mode address. In this document, the real mode address is assumed to be within the context of a compute node's CPU physical address space.
PFMCID	Primary Fabric Manager Component Identifier
PFMSID	Primary Fabric Manager Subnet Identifier
PMCID	Primary Manager Component Identifier
Primary Manager	An in-band Gen-Z manager entity whose scope of control typically is restricted to those Gen-Z components that are part of the same physical assembly or a small set of assemblies in close physical proximity. A Manager must reside on the same subnet as all components it manages as a Primary Manager.
PTE	Page Table Entry. PTEs define a relationship between a Gen-Z fabric address and a local address space associated with a Gen-Z Requestor or Responder function. See Gen-Z Core Specification for details.

Region Key (R-Key)	An opaque object associated with a Gen-Z page. An R-Key is used to validate page access authorization, e.g., to prevent erroneous software or hardware access, and as such, should not be considered a security mechanism. An R-Key may also be used to accelerate address translation. R-Keys are only present on select packet formats.
Region Key Domain (R-Key Domain, RKD)	The upper 12 bits of the 32-bit R-Key value is deemed the R-Key Domain number. Requestors that are enabled to insert R-Key values into certain packets must be authorized to generate R-Keys with a specific R-Key Domain number.
Resource Manager	An upper-layer management entity whose scope of control includes the logical resources such as compute nodes, SCM nodes, FAM nodes, Accelerator nodes, and I/O gateways found on the Gen-Z fabric.
SCID	Source Component Identifier indicates the Component ID (CID) of the source of an explicit OpClass packet.
SCM	Storage Class Memory
SFMCID	Secondary Fabric Manager Component Identifier
SFMSID	Secondary Fabric Manager Subnet Identifier
SFW	System Firmware
SSDT (Single-Subnet Destination Table)	Used by Requesters that support single-subnet communications to look up the allowed egress ports to which a new request may be sent. See Gen-Z Core Specification for details.
SSID	Source Subnet Identifier indicates the subnet ID of the source of an explicit OpClass packet.

Subnet ID (SID)	A value that uniquely identifies a subnet within a multi-subnet topology that is explicitly addressed within a packet.
SCV	Subnet and Component Valid flags: Determine if the corresponding CID and SID fields are configured. See Gen-Z Core Specification for details.
transparent router	A component or component-integrated functionality that performs packet relay between two subnets unbeknownst to the source and destination components.
UEFI	Unified Extensible Firmware Interface
UUID	Universally Unique Identifier as specified in ITU-T X.667. UUID are used to identify components, identify vendor specific capabilities, etc.
VCAT	Virtual Channel Action Table. These tables indicate which Virtual Channels of the Gen-Z fabric are appropriate to use to relay packets. See Gen-Z Core Specification for details.
ZMMU	Gen-Z Memory Management Unit. See Gen-Z Core Specification for details.
Zone ID	Assigned to each Fabric Management Zone for software tracking purposes by the Fabric Director.
Zone Manager	A Primary Fabric Manager in the Management Zone that represents the zone to the Fabric Director and coordinates the other PFMs in the zone.

2. Gen-Z Fabric Management Overview

2.1. Gen-Z Management Layers and Roles

The following diagram illustrates a representative management stack that can be matched to the features and capabilities defined in the Gen-Z Core Specification. Only the composability, resource and fabric connectivity layers, and the APIs that connect them to other layers are within the scope of this document on Gen-Z Management.

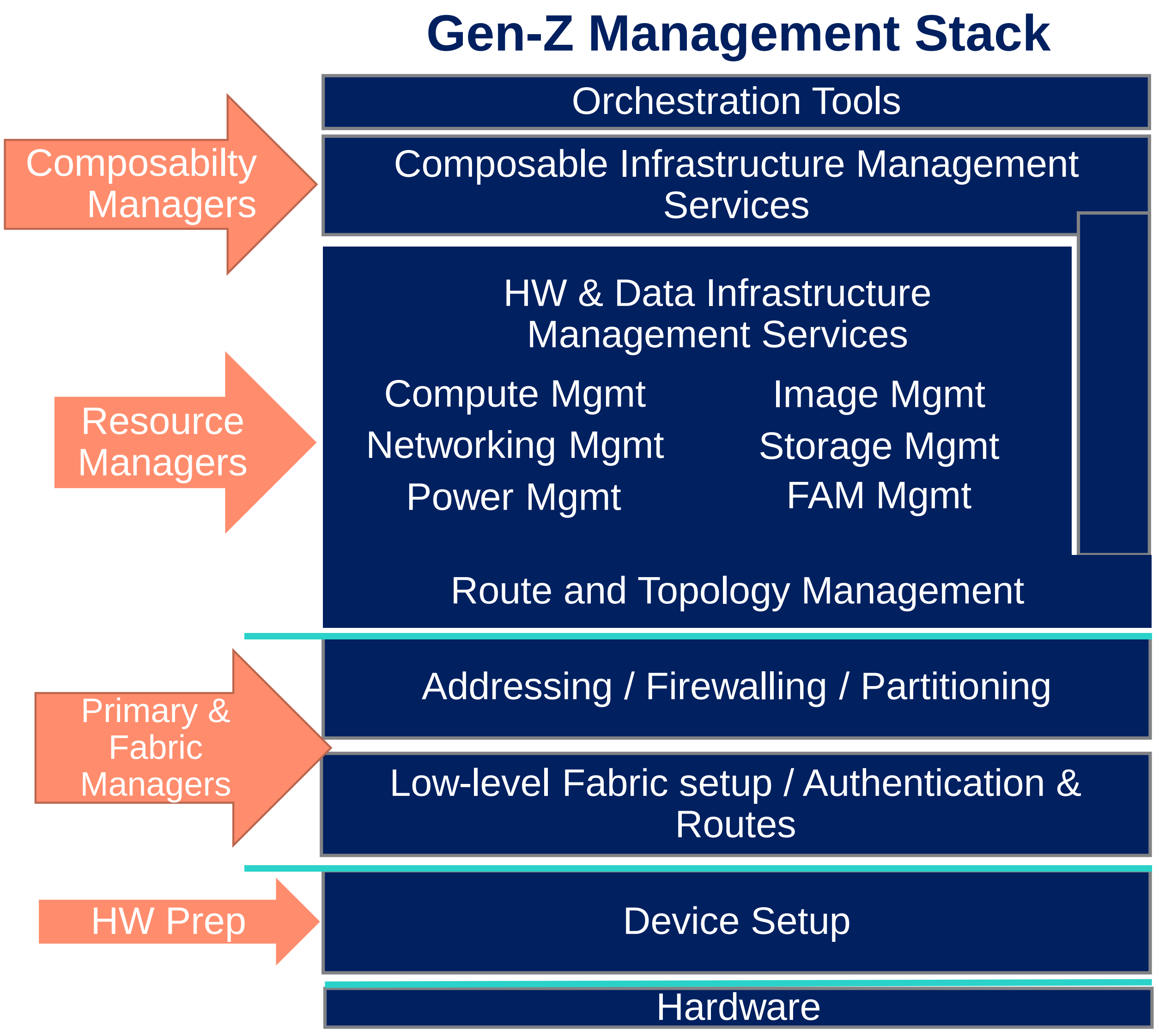

Gen-Z Management Stack
Orchestration Tools
Composabilty Managers
Composable Infrastructure Management Services
HW & Data Infrastructure Management Services
Resource Managers
Compute Mgmt
Image Mgmt
Networking Mgmt
Storage Mgmt
Power Mgmt
FAM Mgmt
Route and Topology Management
Primary & Fabric Managers
Addressing / Firewalling / Partitioning
Low-level Fabric setup / Authentication & Routes
HW Prep
Device Setup
Hardware

Figure 2-1: Gen-Z Management Software Stack

The descriptions and classifications of the management stack layers are as follows, from bottom to top:

Table 2-1: Gen-Z Management Software Stack Descriptions

Scope	General Responsibility	Description
HW Prep	Device Setup	Low level component setup operations such as asserting power, clocks, and resets. Establish component hardware defaults, additional initial

		state, and link states as required by the platform to prepare components for being fully configured by an in-band Gen-Z manager.
Primary & Fabric Manager	Low-level Fabric Setup / Authentication and Routes	Establishing physical connections among components.
	Addressing / Firewalls / Partitioning	Enabling specific endpoints to access specific logical resources using the available physical connections.
Resource Managers	Data Infrastructure Management Services	The general name of the collection of services that manage the persistent data and the logical (data) address space of the Gen-Z fabric.
	Hardware (HW) Infrastructure Management Services	The general name for the collection of services that manage physical resources and their lifecycles.
Composability Managers	Composable Infrastructure Management Services	A central management toolset that abstracts the layers below it into pools of resources and establishes logical servers/platforms and binds them to data structures, other servers, accelerators, networks, boot images, storage and persistent main memory.

2.1.1. In-band vs Out-of-Band Management

The following definitions apply within this document

- **In-band Management:** making changes to a component's control space using CTL-Read and CTL-Write packets (in P2P topologies using P2P OpClasses) or Control Read and Control Write packets launched into the switched fabric by a Gen-Z Requestor (using Explicit OpClasses).
    - Any management thread of execution on any processor of any ISA that can cause any of the control structures to change on some component at some endpoint on the fabric by way of launching a control space packet into the fabric is capable of being an in-band manager.
    - P2P connected components only support a directly attached manager, and do not support access controls such as R-Keys and Access Keys. Therefore they are not discussed often in this document, which is focused on the in-band management of general, composable resources accessible by multiple remote hosts.
- **Out-of-Band Management:** making changes to a component's control space using read and write operations without launching control space packets onto the fabric through a Gen-Z Requestor.
    - Any management thread may opt to use OOB management paths to control components which are accessible in this manner. Such access methods are platform and component specific, so the plug and play compatibility among managers could be significantly impacted when OOB management is used.

- **Hardware Prep (HW Prep):** the Gen-Z management phase that occurs between the end a component's HWInit phase (indicated when Component C-Status HWInit Valid bit == 1b) and the point in time when at least one Gen-Z link attempts to auto-train and connect the component to a Gen-Z fabric.
  - To prepare some components to be ready for configuration via in-band management, some local OOB management capability may be required after the end of HWInit.
  - At a minimum, all link interfaces which are expected to auto-train when the components at both ends of a link are powered up, shall be initialized for such auto-training via hardware initial conditions or via out-of-band means in a platform and component specific manner during HW Prep phase.
  - Thus, this document defines the value of certain Gen-Z control structure fields and component states at the end of the HW Prep phase and these conditions may be different from those left on the component by full or partial resets and HWInit phases of component bring up.

2.1.2. Manager Roles and Related Definitions

The following definitions apply to these terms within this document:

- **Primary Manager:** an in-band Gen-Z manager entity whose scope of control is typically restricted to those Gen-Z components that are part of the same physical assembly or a small set of assemblies in close physical proximity (such as a single enclosure or simple rack). A Primary Manager is usually in charge of Gen-Z components that are not shared beyond this small physical scope. The Primary Manager's ID is found in the PMCID field of the Core Control structure.
  - The Primary Manager CID has no corresponding subnet ID (SID), so all Primary Managers must reside on the same subnet as any components they manage.
- **Fabric Manager:** an in-band Gen-Z manager entity whose scope of control includes multiple enclosure solutions such as one or more racks of servers or Gen-Z based SCM. Fabric Managers are usually in charge of Gen-Z components that are shared across multiple enclosures and racks. The Primary or Secondary Fabric Manager's IDs are found in the PFMCID/PFMSID or SFMCID/SFMSID fields of the Core Control Structure.
  - The Fabric Managers may reside on any subnet within the fabric.
  - Fabric Managers that manage fabrics that have only one subnet do not have to use global CIDs (GCIDs).
  - Fabric Manager Requestors that target Responders that do not support GCIDs are prohibited by the core spec from using GCIDs when controlling such components.
  - ***Primary Managers vs Fabric Managers*:** both managers are granted the same permissions by the hardware and fulfill the roles associated with actual control over hardware components, but only one of the two types is recognized as the component's active manager type at any instant of time based on the setting of the Manager Type bit in the Core Control Structure.
  - A Fabric Manager may initially serve as the Primary Manager for the components that make up the compute server which is executing the Fabric Manager code or are local

to the same enclosure. Primary Managers may promote themselves to Fabric Managers when they need to fulfill a more global role.

- **Manager Domain:** all the Gen-Z components currently managed (owned) by a given Gen-Z manager entity. A Manager Domain is not required to have its own unique CID name space; a complete fabric may have several Manager Domains, each of which contains a subset of the total components in the complete name space.
- **Resource manager:** an upper-layer management entity whose scope of control includes the logical resources such as compute nodes, SCM nodes, FAM nodes, Accelerator nodes, and I/O gateways found on the Gen-Z fabric. These managers control logical abstractions, not just physical components. Often these logical abstractions are mapped to the component's data space. For example, a FAM resource manager would track all free space in a FAM array, and be the librarian-like manager of all persistent data objects stored in any of the persistent FAM modules. Such a FAM resource manager would establish which Requestors are allowed to read or write which data objects (address ranges) within a shared pool of SCM media controllers. Resource managers likely call into a Fabric Manager and have the Fabric Manager enable specific connections and permissions in the hardware control space. Some resource managers may be given direct control over sub-regions of control space on specific components, particularly those sub-regions that control data space resources of those components.
    - Legacy 'Fabric Managers' such as a Virtual Connect manager or an Infiniband Manager are abstracted away from actual hardware control by making them 'resource managers' offering networking and messaging services on top of the Gen-Z fabric.
    - A Gen-Z fabric may co-host messaging and networking traffic at the same time as it passes storage and memory accesses. The 'routed topologies' for each type of traffic may be different.

- **Composability Manager:** an upper-layer management entity whose responsibilities include
    - Initial ownership of all Gen-Z composable resources

    - Keeper of the 'grand plan' that guides the binding of specific resources (all or part of specific components) to resource managers (for pool use) or to individual logical servers for OS and application use.

    - Establishes the Fabric-UUID which is the handle or name of the GCID Namespace

    - Enforcing security policies established by the grand plan architect. Proposals for connections between endpoints coming from multiple resource managers or spread across time each have to be verified as adhering to all the tenant domains and protection policies.
        - Example: If a networking manager decrees that Host A have a messaging connection with Host B, the security analysis features need to be in place to prevent Host A from attempting to access FAM that is to be accessible only by Host B, even if some other portions of that FAM is used for said messaging buffers.
    - Coordinating the creation of R-Key Domains and Access Key domains and the allocation of resources to those domains.
        - R-Key Domains and Access Key domains manifest themselves as Redfish fabric 'zones'. Allowed values or ranges of values for the Redfish fabric zones

are established by the Composability Manager. See the Redfish core specification at https://www.dmtf.org/standards/redfish for details.
    - The Composability Manager tracks the actual R-Key Domain and Access Key values assigned to resources, but it does not create them.
- **Route and Topology Management:** An upper-layer resource management entity whose responsibilities may include
  - Impressing the logical topology chosen by the Gen-Z fabric administrator onto the Gen-Z fabric's physical connections
    - For example, a Resource Manager may create a strict tree structure between itself and an array of media controllers while enforcing a simple star fabric among media controllers and servers accessing the media in the data space.
    - For example, a networking manager may create a hyper mesh or a CLOS tree among all servers for messaging and network functions.
  - Assigning the actual sub-net ID (SID) and component ID (CID) of components based upon where in the logical or physical connection space they reside.

- **Management Zones**
  - Gen-Z **Management Zones** are a named logical collection of Fabric Manager Domains.
  - Management Zones are given a zone ID for software tracking purposes by the Fabric Director
  - Management Zones are logical collections, not physically grouped collections such as a sub-net.
  - These collections of manager domains are created to improve the efficiency of management services that are applied to large scale fabrics with large numbers of Fabric Managers.

- **Fabric Director**
  - A **Fabric Director** is a manager service that creates hierarchical management zones and is the coordinator (commander) of all the Fabric Managers in every zone.
  - A Fabric Director may be co-located with and be contained within the same executable as a Fabric Manager, but is a separate logical Gen-Z manager service.
  - A Fabric Director may be co-located with and be contained within the same executable as a Composability Manager, but is a separate logical Gen-Z manager service.

- **Local Node Services**
  - Those Gen-Z management services which are executed by hardware or software for applications, OS, BIOS, or node managers on an endpoint server or node. Local Node Services are not necessarily part of a Gen-Z manager and are not required to execute as part of a Primary Manager or Fabric Manager thread. If Local Node Services are executed independently of a Primary or Fabric Manager, they may communicate with a Fabric Manager or a Primary Manager using explicitly addressed Control Write Messages across the Gen-Z fabric or via any standard IPC method outside the Gen-Z fabric.
    - Local Node Services may also communicate with the local platform's BMC (or equivalent) to exchange data or actual messages with a Fabric Manager or Primary Manager via out-of-band methods.

- **CID Namespaces:**
    - All the Gen-Z components that physically connect to a Gen-Z fabric must be assigned a unique, valid Component ID / Subnet ID value pair to establish its identity on the fabric. The collection of all such possible CID/SID values, and whether they are currently assigned or not is the CID Namespace for the given Gen-Z fabric. If two physically connected Gen-Z subnets are logically prevented from transmitting packets to or from one another, these two subnets are not considered part of the same Gen-Z CID Namespace and are logically independent fabrics. These two fabrics would have different Fabric-UUIDs.

- **Manager Domain Collision:**
    - A situation wherein one or more Gen-Z links have components at each end that are each claimed by different managers. Thus, these links straddle different manager domains.

2.1.3. Gen-Z Component Ownership

The above roles of the Gen-Z management stack imply that either a 'Primary Manager' or a 'Fabric Manager' entity is normally in control of the component's Core Control Structure. Such a manager can acquire control over a component's control structures using some or all of the following methods:

1) The PMCID or PFMCID fields can be filled in by capturing the SCID included with the first Control Write Packet *received and executed* by a component while it is in the C-CFG (component configuration) state and has no valid manager established. See section "8.6.1 In-band Discovery and Initialization" of the Gen-Z Core Specification for complete details.
    a. NOTE: This 'capture' feature does not apply to control space *reads* of an un-managed component.
2) An existing manager (in-band or out-of-band) can write the CID of the new in-band manager into one of the in-band manager CID fields [PMCID | PFMCID/PFMSID | SFMCID/SFMSID], set the appropriate valid bits and adjust the 'manager type' bit accordingly. See Section *4.3.1 Explicit Ownership Transfer* for details.
3) Most of the component control fields can be set to a valid value after cold reset or power up reset in a platform and component specific manner as part of the HW Prep phase. This is true of the PMCID, PFMCID/PFMSID, and SFMCID/SFMSID fields and their valid flags.
    a. The component can obtain details of desired post power-on values using component-specific means such as serial ROM reads, strapping pin sampling, or hardwired values in the logic.
    b. The component can be given post power-on and reset values using platform specific out-of-band management paths.
4) An existing manager can reset the component and leave it for another manager to discover and assume control using first control write SCID capture, as in option 1.
    a. The existing manager can use some or all of the 'Software-defined Management' sticky bits in the Core Control Structure to indicate to itself or to other managers that this component has been reset for a specific reason.

b. The existing manager can use some or all of the Interface Software Defined I-Bits to indicate to itself or to other managers which interfaces connect to this manager's domain.

2.2. Gen-Z Management Responsibilities

In general, Gen-Z uses a software defined management model to keep the hardware simple and efficient. The Gen-Z Core Specification defines the many actual control fields within the hardware control structures of a component, but does not define the software policies and command sequences that are required to set up and operate a functioning Gen-Z fabric using such components. The first step in defining the management software is to define what features and functionality the software stack must deliver.

The following is a (non-exhaustive) list of the principle roles fulfilled by a Gen-Z management subsystem:

- Planning the GCID Namespace and Topology
- Discovery and Enumeration
- Component and Subsystem Integration
- Allocating, Binding, and Partitioning of Gen-Z Resources
- Resiliency Support
- Dynamic Component and Dynamic Fabric Management
- Audit Logging, Errors and Events Management
- Security
- Power Management

The following sub-sections offer informal definitions, some specific requirements, and examples of these roles.

2.2.1. Planning the Namespace and Topology

Each manager instance launches with a basic idea of its scope and purpose for its domain of control, the classes of topologies it is expecting to discover, and the possible end-states for its domain of control.

- Examples of initial scope and purpose:
  - A Single Server with a Primary Manager is anticipated to always be a stand-alone node and never join a fabric and never connect to another Gen-Z network or subnet.
  - A Single server with a Primary Manager is anticipated to prepare the local node to eventually merge with a larger fabric which is under control of a more global Fabric Manager. The Primary Manager may have a pre-assigned CID namespace that is expected to be compatible with the Fabric Manager's namespace. The Primary Manager expects to negotiate the transfer of its managed components to the control of the Fabric Manager.
  - A Fabric Manager node is expected to acquire control over all shared fabric switches and endpoints found now and in the future on the entire fabric.

  - A Fabric Manager node understands it is restricted to managing a subnet or fabric zone, and that it must merge its manager domain with the greater fabric and negotiate its role with a Fabric Director or federate its Fabric Manager role with other Fabric Managers.

Every manager instance will start with either an explicit plan, or will adopt a standardized set of defaults so that basic policies can be applied to the decisions it will need to make as it fleshes out its manager domain. Example decisions a manager will need to make:

- How does the manager choose which Component IDs to assign to which (if any) of the various Gen-Z components it discovers?
- How many 'hops' down a chain of Gen-Z linked components does the manager explore?
- What does the manager do when it discovers a component on a different FRU?
- What does the manager do when it encounters a component already claiming to be managed by a different manager?

Every manager has an algorithm for managing its namespace and generating CIDs, and policies in place for choosing which branches to explore on the Gen-Z topology before it ever starts the 'Enumeration and Discovery' process.

For a Primary Manager instance, it is acceptable to have no plan at initial boot and therefore require admin intervention (via out-of-band means such as an Ethernet Portal or system console) before proceeding.

2.2.2. Enumeration & Discovery

A major duty of a Gen-Z manager entity is to determine which Gen-Z components are accessible by the manager and which are in need of configuration and management by this entity.

- **Discovery** is the process of detecting the presence of a Gen-Z compliant component upon the fabric, and establishing a management connection with the component.
- **In-band discovery** shall be supported, such that no out-of-band or side band signals are required between a manager node and any component to alert the manager to the presence of that component on the fabric. In band discovery therefore assumes that the presence of a component on the fabric is evidenced if and only if a link to that component is in the L-up state.
  - A component is not known to be present and is not discoverable by an in-band manager if it is unpowered or it has no active links (no links in L-Up state).
  - Out-of-band methods such as 'presence pins' can indicate the physical presence of a component even if that component is not powered or not reachable via a Gen-Z link. Such out-of-band discovery techniques are not within the scope of this specification.
  - Out-of-band or platform / vendor specific methods may be necessary to enable specific component links which are tied to 'fabric managed' peers to auto-train and enter L-up.

- **Enumeration** is the act of collecting relevant information about discovered components and creating a master list of such entities and their associated meta-data.
- **In-band enumeration** shall be supported, such that no out-of-band or side band signals are required between a manager node and any component to enable the manager to determine the component's base class, supported capabilities, and current operational states.
    - The Core Spec defines an optional FRU-UUID which is a unique vendor assigned UUID that is given to all components of a specific FRU instance. Multiple FRU instances shall have unique FRU-UUIDs. See Core Structure in Table 8-3 of revision 1.0a of the Core Specification for details.
        - An in-band manager may obtain the FRU's class and capabilities from some external cross reference to the FRU-UUID.
        - Further, if implemented by all components on a given FRU, the FRU vendor will program the same FRU-UUID into each component so that Managers can discern which components reside on which FRUs.
    - In-band enumeration capabilities defined in the current 1.0 revision of the Gen-Z Core Specification apply only to Gen-Z in-band managed components. Additional non-Gen-Z resources that are co-located on the same FRU are not directly visible to a Gen-Z manager.
    - This specification recommends Redfish and MCTP compatible methods and schemas be supported so that Gen-Z managers can exchange information with non-Gen-Z managers.
- Enumeration also implies the tracking of discovered components and meta-data associated with them. **The CID Namespace** of a manager domain shall be maintained by the manager. The manager shall assign and track all Component IDs (CID and SID) and associated meta-data of components managed by itself on all subnets visible (reachable) from the manager's interface into the fabric.
- **Configuration** is the process of setting the control structure values of a component to place it in a state where it is ready for handoff to the next layer in the management stack, or where it is ready to be activated (moved to C-UP state).
- **In-band configuration** of discovered components shall be supported, such that a Primary Manager or a Fabric Manager shall be enabled to establish the desired state and configuration settings upon any discovered components using only in-band methods, and thus have no required reliance upon out-of-band or sideband signals between the manager and the managed component.
    - A key requirement here is that an in-band manager must be able to affect a full component reset of any component without invoking any out-of-band signaling. Any forms of resets generated in-band may have vendor or platform specific side effects. Any such side effects shall not prevent an In-band Manager (the same or a different manager) from regaining control of the component by following the policies architected herein.
    - Another key requirement here is that no FRU or component will be designed such that any In-Band manager can destroy persistent data (in non-volatile

media) behind a Gen-Z compliant media controller by the intentional or un-intentional assertion of an in-band full component reset. This requirement is defined in the Core Spec under 12.11., Full Component Reset, and just repeated here for clarity.

2.2.3. Allocating, Binding, and Partitioning

Once a functioning Gen-Z fabric has been inventoried and placed under a management scheme, and the CID namespace is defined and filled in, resources need to be bound together into functional entities such as logical servers, logical storage appliances, and gateways to I/O fabrics and external networks. The various Gen-Z manager roles (Composability Manager, Resource Managers, and Primary Manager or Fabric Manager) may divide or consolidate the many necessary and optional functions which make up a Gen-Z ecosystem in many ways.

By definition herein, however, the Fabric Manager *role* (be it run as a Primary Manager or a true Fabric Manager) is the role which has direct Gen-Z packet access to the Gen-Z components' architected Core Control structures.

- **Allocation** of Gen-Z resources is the act of associating specific Gen-Z resources with specific Resource Managers, compute platforms, or user solutions. Resources thus allocated are considered to be reserved for those services, pools, platforms or solutions.
- **Binding** of Gen-Z resources is the act of creating logical and physical connections among such resources and the Resource Managers, compute platforms, and user solutions to which they are allocated. The Gen-Z Management stack will need to track both the logical and the physical resources, their allocation, and their bindings.
- **Partitioning** of Gen-Z resources is the act of tracking which allocations and bindings are allowed, and which are not allowed, creating the allowed paths and connections and installing the firewalls which prevent the dis-allowed paths and connections.

The following section outlines the management activities and responsibilities associated with allocating, binding, and partitioning duties as envisioned for each of the major roles of Gen-Z composable fabric management.

The Composability Manager role will typically be responsible for

- Collecting the global Gen-Z resource inventory as a Redfish Gen-Z resource tree
- Tracking resource health and status
- Collecting metrics and adjusting QoS settings
- Maintaining and distributing the global Gen-Z CID Namespace
- Creating the Fabric-UUID and attaching it to all descriptions of the CID Namespace
- Creating and maintaining the Gen-Z Services directory
- Allocating physical resources to Resource Managers
- Composing and launching logical platforms
- Creating, merging and tracking single and multi-tenant partitions
- Creating and tracking A-Key domains and A-Key values
- Tracking R-Key domains and R-Key Domain values, as well as specific full 32-bit value R-Keys in use by the Gen-Z management stack
- Validating requests for connections and paths, and associated access rights
- Maintaining the list of bound resources and free resources

    - Including the list of fine-grained resources protected by R-Keys
- Creating and handing down Redfish descriptions of accessible Gen-Z resources as bound to booting or running platforms
- Calling the Fabric Manager role to update actual Gen-Z hardware settings

The Resource Managers are the roles typically responsible for

- Requesting and accepting the appropriate type of Gen-Z physical resources (as described using Redfish schemas) and integrating them into the resource pools which they manage
- Advertising any Services (EG. storage services) supplied by the Resource Manager (Redfish/Swordfish schema)
- Publishing logical resource pool inventory (again via Redfish/Swordfish/JSON)
- Accepting requests to bind logical resources to logical platforms, mapping them to physical components, and recommending connections and paths among the resulting instantiated instances.

The Fabric Manager role is responsible for the following pieces of the allocation, binding, and partitioning of resources:

- Accepting descriptions of physical paths, connections, and permissions and generating Gen-Z Control Structure changes on the impacted components
- Delivering the Redfish descriptions of the logical resources mapped to a Gen-Z processing node (specifically what is mapped to the Gen-Z Requestors on a given node)
    - Doing this notification strictly in-band will require a minimalistic Local Node Service presence on the target node (probably running as a Gen-Z aware driver in the SFW/BIOS/UEFI and/or as a privileged process within an OS).
- Installing actual R-Key Domain and A-Key values on Gen-Z components
- **The Route and Topology Manager sub-role** is the role responsible for
    - Impressing the logical topology expected or requested by the various Resource Managers
    - Tracking the subnets in use
    - Assigning subnet IDs and component IDs (SIDs, CIDs) to new components added to the fabric based upon the logical topology being used
        - If building the fabric resource tree for the first time, the grand plan will define the policies around subnet use and subnet ID assignment.
        - If no grand plan has defined any subnets or any policies that dictate subnet layouts, the default is to use only a single subnet until it is full.
        - This sub-role may be distributed among the Composability Manager, the Resource Managers, and the Fabric Manager based on the preferences of the software architects. However, the Fabric Manager must create the initial control plane fabric topology, which by default will look like a simple tree structure with the FM at the root and the managed components in the FM's domain as branches (switches) and leaves (endpoints).

2.2.4. Component and Subsystems Integration

Once a given Gen-Z component is discovered, the Gen-Z management system must integrate the component into the Gen-Z fabric environment as appropriate. Integration involves setting up switches, transparent routers, and other components capable of performing packet relay functions to create a 'fabric' with a desired logical topology, and setting up media controllers, bridges, eNICs, and other endpoints to participate as resources on the fabric.

- **Fabric Integration** is the process of setting up the individual components on a fabric to have an appropriate set of connections to each other, with proper paths and permissions established between every end point and potentially every other endpoint. The Gen-Z Fabric Manager layer is responsible for issuing the actual control space reads and writes that establish connections and routing paths, but is not necessarily the manager layer that dictates which possible paths are enabled for which connections among which components. Initial fabric integration is primarily concerned with creating an appropriate set of connections between Gen-Z components and the Primary Manager or Fabric Manager that is doing this stage of integration.
- **In-band fabric integration** shall be supported such that no out-of-band or side band signals are required to properly integrate components per a connection and permissions plan known to the manager entities.
- **(Sub)System Integration** is the process of making logical Gen-Z fabric resources and services available to compute servers and appliances running code unique from the Gen-Z manager code. Examples:
    - OS on a server 'integrates' regions of FAM (fabric attached memory) into its physical address space after coordinating with the FAM resource manager to establish the required mappings and connections within the Gen-Z fabric.
    - A storage appliance 'integrates' regions of SCM (storage class memory) into its physical media pool after coordinating with the SCM resource manager to establish the required mappings and connections within the Gen-Z fabric.

  System integration involves conversations between high level resource managers which then result in requests to Gen-Z managers to make the actual hardware changes. These high level conversations are not strictly required to occur in-band across Gen-Z, but this specification will outline policies and mechanisms that will enable such communications to be entirely in band.

  The physical topology of the Gen-Z fabric is dictated by the physical presence of Gen-Z links via cables, PCB traces, or possibly wireless transceivers. However, the Route and Topology manager sub-role will enable a logical topology (EG. CLOS tree, Fat tree, or N-dimension hyper-mesh) by enabling specific routes (paths) between specific endpoints.
- **Advertisement of Gen-Z Resources and Services** to applications and OS is a run-time function of the Gen-Z management subsystem. Communications between Gen-Z Fabric Managers and System Integration managers such as BIOS, OS, or even user applications should rely on existing communication mechanisms and platform integration schemas and standards such as UEFI, ACPI, SMBIOS, etc. where ever possible.

2.2.5. Resiliency Support

Various versions of commonly available redundancy and resiliency tactics are expected to be available from as robust a standard as Gen-Z. Not only should multiple paths between connected endpoints be available for the sake of fabric redundancy, but the Fabric Manager itself should not be a single point of failure.

- **Redundant Fabric Managers:**
    - The Gen-Z Core Specification defines both a Primary Fabric Manager and a Secondary Fabric Manager, each of which is awarded full and equal control over all of a component's Core Control structures when the component is enabled for in-band management and the Manager Type bit is set to 'Fabric Manager'.
    - There are many existing methods of keeping redundant managers in sync with each other, but in this architecture there are some requirements and some recommended practices defined to enable inter-operation of multi-vendor managers:
        - The Primary Fabric Manager and the Secondary Fabric Manager shall use the same MGR-UUID.
        - The Primary and Secondary Fabric Managers shall support the mandatory message types and associated payload formats for in-band manager to manager communications as defined in Sections 4.2 *Manager to Manager Communication* and *Section 5.3 Manager to Manager Messages.*
        - When using in-band Gen-Z packets to perform manager to manager communications between the Primary and Secondary Fabric Managers, the messages shall be sent as Control Write MSGs or as multicast Control Write MSGs (if multicast is supported).
        - The Primary and Secondary Fabric Managers may use any appropriate manager to manager messages defined in Section *5.3 Manager to Manager Messages* passed via any out-of-band mechanism.
        - The Primary and Secondary Fabric Managers may use any in-band or out-of-band transport mechanism to exchange heart beats, messages, and critical state. Such transport mechanisms other than Gen-Z are beyond the scope of this document.
    - When a Primary or Secondary Fabric Manager fails and support duties fall to the alternate Fabric Manager, the components within the associated manager domain are not aware of this transition. Therefore components in an operational state (C-UP, C-LP, C-DLP) will maintain their current state and continue with operations as normal.
        - The process of failover to the alternate Fabric Manager shall not disrupt the ongoing operations of the managed components in that manager domain.
        - The recovery or replacement of the failed Fabric Manager shall not disrupt the ongoing operations of the managed components in the manager domain.
        - Should there be no alternate Fabric Manager available, the manager domain may continue to operate as normal as long as possible.
            - To prevent service disruptions, when a replacement Fabric Manager becomes available, it should be given the identity (CID, SID, MGR-UUID) of the failed management entity.
            - If this identity cannot be cloned, the behavior of the manager domain is un-defined, and the recovery of these components is beyond the scope of this document.

  - Additional manager transition scenarios beyond the above Secondary and Primary Fabric Manager co-owner scenarios are discussed in Section 4.3.1, *Explicit Ownership Transfer*.

- **Multi-path Enablement and Management:**
  - **Fabric Path Redundancy**: Exists if a request packet may travel over more than a single choice of egress link from at least one component in its journey to the targeted Responder. If all components between the Requestor and the Responder have multiple paths available, then full fabric redundancy is available.
  - **Component Path Redundancy**: Exists if a request packet may egress and ingress the component on more than a single Gen-Z link, *and* there is full or partial fabric path redundancy to the ingress links.
  - The Primary Manager and the Fabric Manager may enable more than one Gen-Z path to a managed component over which the manager may send control space packets. Primary and Fabric Managers should enable multiple paths for control space packets between themselves and their managed components as soon as possible after the component is enabled for C-UP.
  - Multiple paths between a non-manager Requestor and a given Responder may also be enabled to create a redundant data space fabric. The strategies for such data space redundancies are part of the responsibilities of the Route and Topology Manager, and are beyond the scope of this document.
  - The multiple path and multiple route capabilities of Gen-Z are controlled by the Route Control Structure, the Component Switch Structure, and the Component Destination Table Structure of the various Requestors, Responders, and Switches within the fabric.
- **Fault Tolerant Fabrics:** Combining Component Path Redundancy with full Fabric Path Redundancy and end-to-end unicast packets, it is possible to create a form of Fault Tolerant Fabric wherein no single fabric switch component or single fabric link failure can prevent the eventual transfer of a packet from Requestor to Responder.
  - However, there are many restrictions on time-outs, forward progress screens, and use of common components and links on enabled redundant paths that must be enforced to craft such a fault tolerant fabric between endpoints. Therefore, how to manage fault tolerant Gen-Z fabrics is beyond the scope of this document.

2.2.6. Dynamic Component and Fabric Management

To support the previous requirements for managing partitioning and resiliency within a shared, composable Gen-Z fabric, a Gen-Z manager will make dynamic changes to components (resources) and the connectivity (routes and permissions) of the fabric. Dynamic features can also be offered on smaller, stand-alone Gen-Z installations with no shared resources by making appropriate optimizations.

  - **Fabric Hot Plug support** is dynamic addition and removal of Gen-Z components to or from the fabric without disturbing other entities on the fabric or traffic flowing among them. Gen-Z managers shall support fabric hot plug of Gen-Z components, though the methods used to do so are not defined in this specification.

If the components being added or removed are currently in use or to be used by processing threads currently running on one or more nodes, those threads should support Dynamic Resource Add or Delete, per below.

- **Dynamic Resource Add or Delete** is the logical acquisition and/or release of Gen-Z resources by a processing thread during a time period where other Gen-Z resources are already in use by the same or another thread. The Gen-Z managers shall support communication mechanisms of Chapter 4 *Dealing with Multiple Managers* to coordinate such changes with those threads or their controlling OS.
- **Dynamic Relay Table Updates and Dynamic Permissions Changes** are changes made to routing and permission controls within the Gen-Z switch and endpoint resources while allowing traffic to remain flowing along un-affected routes or between un-affected endpoints. The Gen-Z Fabric Manager shall make such changes without forcing un-affected traffic to be suspended.

  Gen-Z components which support dynamic changes shall support updates to routing and permissions tables affecting routes between two given endpoints which do not disrupt other traffic between either of these endpoints and any other entities with which they have active connections.

2.2.7. Audit Logging, Errors and Events Management

The Primary or Fabric Managers have access to (and therefor own) all HW control structures of a component. Only the most trusted management layer is allowed to alter the critical fields of a component's control space. Whenever another manager role, say a Composability Manager needs to impose a change on a component's control structure, it must ask the Primary or Fabric Manager to proxy the change for them.

However, there are some management roles beyond the scope of the Primary and Fabric Managers that do not lend themselves well to this proxy method.

Events and errors signaled by Gen-Z components often need to be directed to manager roles other than the Fabric Manager or Primary Manager. Finally, there are control space structures that are more or less dedicated to features or functions that are not typically within the scope of the Primary or Fabric Managers. Examples of these include power management, media management, and mechanical structure management.

Named Specialty Managers: Processes associated with component CIDs that match named manager ID fields in the Core Control structure. These named managers share control with the Primary and Fabric Managers over specific features or functions and some related control space structures. The list of such Named Specialty Manager IDs:

- Error MGR CID: a manager CID to which Error notifications may be issued
- Event MGR CID: a manager CID to which Event notifications may be issued
- PWR MGR CID: a manager CID which may issue certain power control opcode packets
- MECH MGR CID: manager CID to which mechanical structure notifications may be issued
- MEDIA MGR CID: a manager CID which may access the Component Media Structures

In addition to the above named managers that receive special privileges within control space, or can be sent classes of notifications, there are other management roles that might be associated with specialized areas of control space or data space. Examples include performance monitoring and application audit logging. Auditing, Logging, and Performance Monitoring are management roles allocated to various layers of the management stack, mostly above the Fabric Managers, or even to application specific threads that may run on non-manager nodes.

Therefore, the Gen-Z Core Specification has architected the Component Core Structure in segments, most of which can be located in different pages of the component's local Gen-Z address space. Further, any given page of a component's Core Space can be tied to a unique Read/Write or Read-Only R-Key using the Component C-Access Structure defined in Section 8.35 of the Core Spec. Any Requestor that can send Control Read and Control Write packets to a given destination with the matching R-Key will be granted access on a page by page basis.

Limited Scope Managers: Processes with access rights to select R-Key protected Gen-Z control structures, but not the full permissions granted to the Primary or Fabric Managers.

- Limited Scope Managers may be threads running on the same host component as a Primary or Fabric Manager threads.
- If Limited Scope Managers have the same CID as a Primary or Fabric Manager, they shall not have the same MGR-UUID (since that would make them indistinguishable from Primary or Fabric Managers)
    - See Section 12.9, Packet Validation and Processing of the Gen-Z Core Specification
    - See Host Manager MGR-UUID Enable field in the Core CAP 1 Control structure in Section 8.15.4 of the Core Specification for information on how the host bridge Requestor chooses a different MGR-UUID to include in packets launched on behalf of Limited Scope Managers when such managers have the same CID as a Primary or Fabric Manager.

The methods to select and designate Named Specialty Managers and assign R-Keys to Limited Scope Managers are beyond the scope of this document.

General Capabilities and Requirements of Audit Logging, Errors, and Events Management

Regardless whether auditing, logging, errors or event management functions remain with the Fabric Manager or are farmed out to one or more Limited Scope or Named Specialty Managers, the Gen-Z Management Architecture defines the features and requirements associated with audit logging, errors and events management in Section *o*
Gen-Z Management Basics

Assumptions and Requirements
The Gen-Z management architecture assumes the following interpretations and requirements:

2.2.8. Security

The Core Specification has an entire chapter (Chapter 16) on the threats, agents, and mitigations of classic Security for fabrics. In addition, there are several industry wide efforts to standardize necessary 'security' functions such as authentication, validation, and attestation across fabrics. So this document is not going to address these same topics in depth. Security discussions in this document are limited to the following topics:

- **Component Authentication** needs and our recommendations about meeting those needs
- **Fabric Partitioning** abilities defined in the Core Spec, some non-obvious associated security concerns, and policies we recommend to ensure Fabric Partition Integrity
- **Domain Boundary Security** policies needed to prevent accidental or malicious traffic from obtaining ingress via links initially connected to non-existent components, unpowered components, unrecognized components, unmanaged components, foreign managed components, or components not within same CID namespace.

Component Authentication:

Every component that has physical access to a Gen-Z fabric is an attack vector. Even if the physical access to the fabric is tightly constrained, to the point that a single most-trusted person is the only one allowed to select member components and personally attaches them to the fabric, there are significant risks that the component is already corrupted by the time it reaches their physical control.

The Gen-Z Core Specification outlines the industry standard component authentication scheme being adopted for Gen-Z use. See Gen-Z Core Specification (Section 16.6 Component Authentication) for references to the applicable standards, a brief overview of the chosen process, and the specific options of the standard to be supported by compliant Gen-Z components.

This component authentication scheme will form the foundation for effective runtime isolation and integrity schemes described below.

Fabric Partition Integrity:

The Gen-Z management stack is the final authority on permissions and access rights between specific Requestors and Responders. (See *Section 2.2.3 Allocating, Binding, and Partitioning* for the definition of Fabric Partitioning.)

The integrity of the partitioning choices is enforced by hardware implementations and management SW policies that see to the integrity of the individual component's control and data access points. The Fabric Manager is the only entity acknowledged by the hardware as having authority (via in-band methods) to alter critical controls such as routing tables, peer access (permission) tables, and link packet filters. This document defines certain requirements upon the Fabric Manager's actions and hardware prep operations to define a base level of integrity for partition choices.

The goals for Fabric Partitioning are to enable:

- *Any Responder to deny access* to data or control structures by Requestors not authorized to have such access.
- *Requestors to launch only the specific request actions (opcodes) to the specific Responders* (and the specific sub-ranges therein) for which they have been granted access by a Fabric Manager.
- *Packets to traverse the fabric between Requestor and Responder using only specifically enabled paths* so as to limit shared links between isolated partitions and to deliver the desired Quality of Service within and among the partitions.

These goals are achieved by tightly controlling the connections between individual Requestors and Responders, and the routes packets follow between them on the Gen-Z fabric.

Connections

Connections between Requestors and Responders require setting up Requestor source permissions, Responder destination permissions, and valid routes between the two components.

Requestors and Responders communicate using two distinct address spaces when addressing specific functionality within each component:

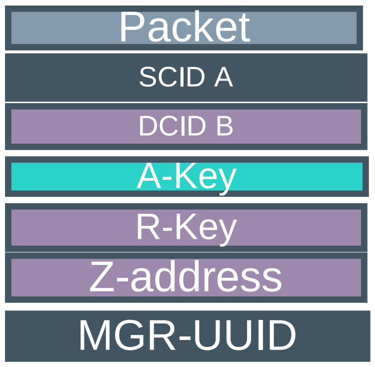
- Data Space (64 bits of component-relative addressability)
  - Accessible in-band via any of the Explicit OpClass packet types except Control OpClass packets
- Control Space (52 bits of component-relative addressability)
  - Accessible in-band via Control OpClass packets

Requestors and Responders communicate across the Gen-Z fabric using two different modes of component ID assignment

- Single Subnet mode, where only the 12-bit Component ID (CID) is needed to address any component
- Multi-subnet mode, where an additional 16-bit Subnet ID (SID) is concatenated onto the front of the CID to create a 28-bit Global Component ID (GCID)

Important Gen-Z packet structures that enable Gen-Z fabric partitioning

The following abstract diagram of a generic Gen-Z packet discloses the central features that enable Gen-Z Requestors to access specific Gen-Z Responders or sub-regions within them. Using these features, Gen-Z resources can be composed into functional systems or isolated into independent partitions.



SCID: The Source Component ID is the CID (and if appropriate) SID of the Requestor which originated this packet. *Responder* hardware looks up the Source ID to determine which, if any, privileges this Requestor has with this destination. See Section 12.8.2 Explicit Packet Processing of the core spec for details.

DCID: The Destination Component ID is the CID (and if appropriate) SID of the Responder to which the packet is targeted. *Requestor* hardware looks up the DCID to determine which, if any, privileges this Requestor has with this destination component. In addition, switches use the packet DCID to choose to which egress port to relay the packet. See Section 8.36 Component PA (Peer Attribute) Structure of the Core Specification for details concerning how Requestors establish the requirements to source an Explicit OpClass packet to the given destination for data space packets.

Z-address: This is the linear byte offset within the targeted Responder's address space that fully specifies the starting byte address for packet opcodes that need such. This linear byte offset may be used to specify a specific sub-region, aka a 'page' of the Responder's address space, and that page number may be associated with a Region Key.

A-Key: The Access Key is a 6-bit value that the Requestor has associated with the destination CID (DCID) of the target Responder. The Responder will validate that packets from the Source (based on SCID) are to contain the matching A-Key. Requestors and Responders cannot communicate if they are not using the same A-Key to make their connection.

Further, the A-Key can be checked at the ingress port and the egress port of all switches in the path between the Requestor and the Responder. The path between Requestor and Responder must also be enabled to relay packets with the A-Key included in the packet. See Section 8.16 of the Core Spec for more details. A-Keys are a coarse-grained isolation mechanism; many components may need to share the same A-Key.

R-Key: The Region Key is a 32 bit value that the Requestor has associated with the sub-region the packet is to address on the Responder. The Responder will look up the required R-Key for that sub-region as established by its manager, and compare it with the R-Key found in the packet. If the two values agree, then the packet is allowed. R-Keys thus act as a destination firewall that can prevent Requestors without the correct R-Key from accessing a specific sub-region of the Responder.

The 12 most significant bits of the R-Key are called the **R-Key Domain (RKD)**, and are validated by the Requestor (See Section 3.2.9.1 of the Core Spec) as allowed values.

MGR-UUID: The MGR_UUID is a software created 128 bit UUID that defines the Manager UUID of the Requestor of the packet. It is only used in select control space OpCodes. It is used to disambiguate among multiple potential managers during initial configuration or manager hand-off situations. Also, Responders will use the Manager UUID in conjunction with the SCID to validate that the Requestor is operating from a manager component (correct SCID) with the proper version of the manager software (correct Manager UUID).

For example, when a Primary Fabric Manager enables a Secondary Fabric Manager, the Secondary Fabric Manager must have the same MGR-UUID. Since the Primary Fabric Manager establishes the MGR-UUID, it should inform the Secondary Fabric Manager of the correct value to use to successfully access components as the SFM.

Table 2-2: Packet Fields Important to Isolation and Partitioning

Packet field	Value Comes From	Requestor Permission checks	Responder Permission checks
SCID	Gen-Z Manager	Not checked. Inserted from Control Structure owned by Gen-Z manager	Checked by Gen-Z tables owned by Gen-Z manager
DCID	Application or host OS (SW)	DCID must be allowed by Gen-Z tables owned by Gen-Z manager	Checked by Gen-Z tables owned by Gen-Z manager
Z-address	Application or host OS (SW)	none	May be checked for R-Key match from tables owned by Gen-Z manager
A-Key	Gen-Z Manager	Inserted from Gen-Z tables indexed by DCID and owned by manager	Checked against value from Gen-Z tables indexed by SCID and owned by manager
R-Key	Application or host OS (SW)	Upper 12 bits (RKD) are checked against RDK structure owned by Gen-Z manager	Full 32 bits checked for match from tables owned by Gen-Z manager
MGR-UUID	Manager SW	Inserted from Gen-Z control structure owned by manager SW	Checked against value from control structure owned by Gen-Z manager

Important Route and Permission Structures that support Gen-Z fabric partitioning

Figure 2-2 *Important Control Structures Dictate Requestor Packet Contents* illustrates how the important logic blocks that supply or generate the values for the various Gen-Z packet fields described in the previous section are related to important routing and permission setting control structures. This diagram shows how a host load or store transaction is mapped through a bridge's zMMU logic to produce a corresponding Gen-Z data space Read or Write packet which:

- Targets the associated Gen-Z destination Component ID (DCID) and destination subnet ID (DSID)
- Contains the required 32-bit R-Key value which the Responder will require before validating the request
- Contains the 64-bit Z-address offset within the data space of the destination component
- Contains the required A-Key
- Contains the Requestor's Component ID and subnet ID as the Source CID and Source SID (SCID,SSID respectively)

The Requestor Control Logic is fed various route and permission details from the associated Gen-Z control structures, which are described following the diagram.

Note that the red data flow lines represent data fields that may be controlled by the host threads, and not by the Gen-Z management entity. These values are possibly compromised. Therefore the route and permission details fed to the Requestor Control Logic are only indirectly selected by host controlled values; the Fabric Manager has final say over actual permissions assigned to a request packet.

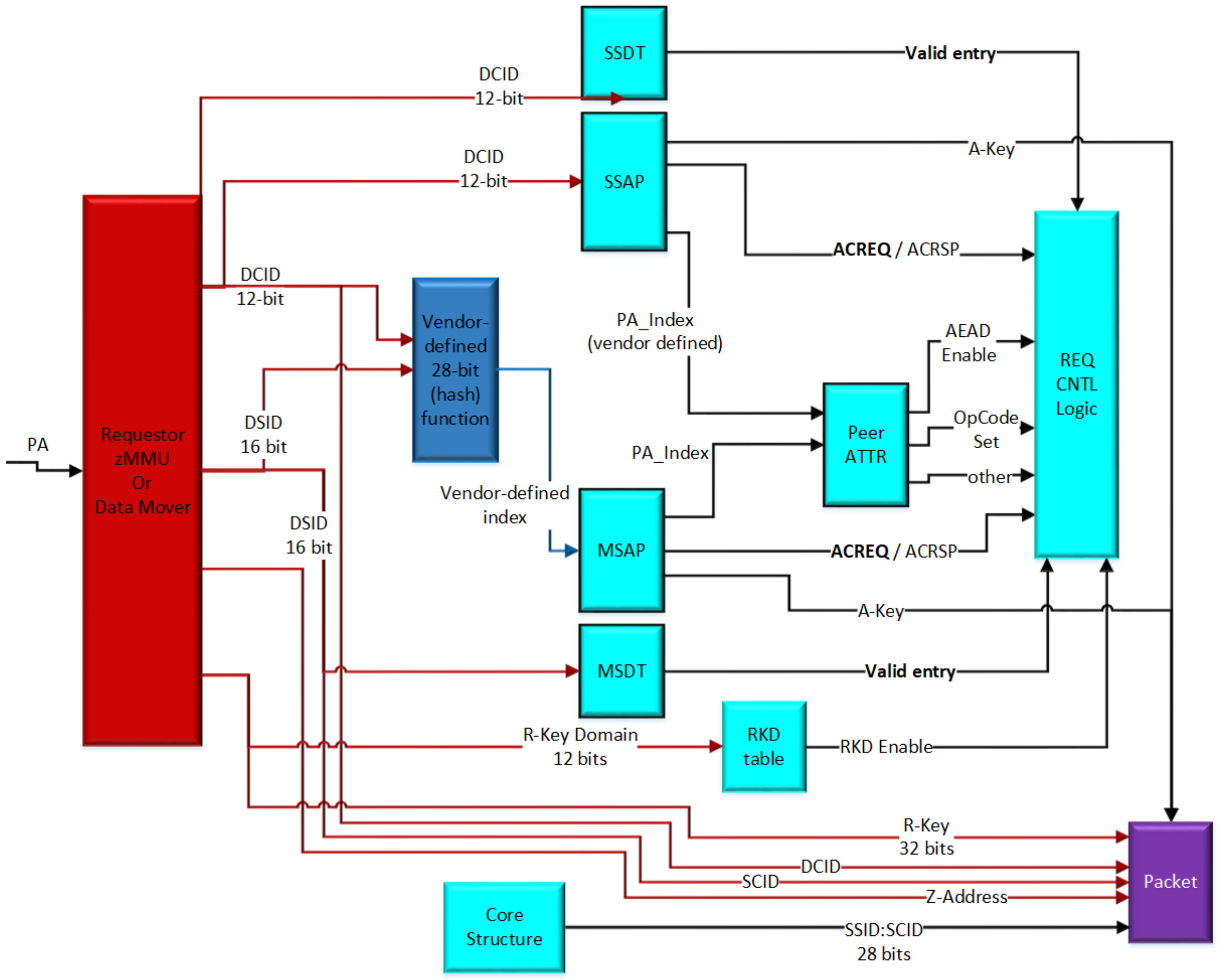


Figure 2-2 Important Control Structures Dictate Requestor Packet Contents

Setting the proper routes and access controls at the Requestor requires the Fabric Manager initialize the following structures:

- **SSAP and MSAP** tables (Single Subnet or Multi-Subnet Access and Permissions tables)
  - These two tables contain indexes into other tables as well as specific access controls
  - The **A-Key** to be used to reach a given destination or group of destination components is set up by the Fabric Manager
  - In a single subnet environment, the 12-bit DCID is used directly as the index into the SSAP table which contains the ACREQ and ACRSP permission fields. Thus, a specific set of permissions can be applied to a specific connection between a SCID and any DCID.
  - However, in a multi-subnet system, the GDCID (global destination CID) is a total of 28 bits of ID. The mapping of the 28 bit GDCID to an index into the MSAP is a vendor-defined function. Hence, many connections may share the same MSAP entry and associated ACREQ permissions. It may not be possible to establish a large set of connections, each with its own

independent MSAP table entry. Destinations are likely lumped into 'component groups' that share common access attributes and permissions with respect to a particular Requestor.

- **ACREQ and ACRSP** (access control – Requestor, access control – Responder) fields of the **SSAP** and **MSAP** tables (See Section 8.36 of the Core Specification)
    - ACREQ and ACRSP establish if the given source of a packet has permission to send packets to the intended destination component. There are three options encoded into the ACREQ or ACRSP fields:
        - This source has no permissions with respect to this destination
        - This source may access regions within this destination using the proper R-Key
            - If R-Keys are not part of the given packet's protection mechanisms, the packet will not require an R-Key check and will be allowed if all other protections are valid.
        - This source may access any region of this destination without generating or checking R-Keys.

The ACREQ and ACRSP access permission fields are only used for data space packets (packets other than the Control OpClass packets). *Control OpClass packets are not subjected to a permission check at the initiating Requestor based on the GDCID of the destination component.*

- **SSDT or MSDT** (routing tables used for packets emitted by the Requestor)
    - A single subnet system will use a simple look-up of the DCID in the SSDT to find one or more egress ports that will conduct a request packet to the desired destination. Any single-subnet DCID can be firewalled off at the Requestor by disabling all egress ports for this DCID.
    - However, in a multi-subnet system, the GDCID is 28 bits, so only the DSID is used to locate an egress port in the MSDT. This collects many destination components together into a group that share a common egress port off the Requestor, so it may not be possible to deny an egress port to any single destination component in a multi-subnet Requestor.
    - If a valid egress port cannot be found in the SSDT or MSDT, the Requestor cannot launch the packet onto the fabric.
    - There are control bits defined by the Core Specification (see Section 8.23 Route Control Structure of Core Specification) which enable more sophisticated look up algorithms to be used with the SSDT and MSDT.
- **Peer-ATTR** (peer attributes)
    - The Peer-ATTR table contains access permissions and parameters for groups of target components. Think of it as setting the access policies for groups of connections.
    - The Peer-ATTR table is indexed by values exported from the SSAP (single subnet access) or the MSAP (multi-subnet access)
- **OpCode Set** structure (found in the Peer-ATTR table)
    - If implemented, dictates which opcodes this Requestor is allowed to send to the Peer-ATTR group to which the destination component is mapped.
    - Specific Requestors may be denied use of specific opcodes when targeting certain groups of Responders.
- **AEAD Enable** (Authenticated Encryption with Additional Data Enable)
    - This is an enable bit that will turn on AEAD MAC protocols for connections to the group of Responders mapped to this Peer-ATTR entry. (see Section 8.35 and Chapter 16 of the Core Specification).
- **RKD** (R-Key Domain Structure)

- The full 32-bit value of an R-Key to include with a packet comes from the host-provided R-Key field of the Requestor zMMU PTE. (See Section 8.40 of the Core Specification.)
- The 12 most significant bits of the R-Key value constitute the R-Key Domain. These host-provided bits are compared to the enabled R-Key Domains enabled by the Requestor's Gen-Z manager. If the manager has not enabled that R-Key Domain, the Requestor cannot launch the packet onto the fabric.

Setting the proper routes and access controls in the switched fabric involves the following structures (and related controls to enable them):

- **LPRT** (Linear Packet Relay Table) and **MPRT** (Multi-subnet Packet Relay Table)
  - The LPRT is indexed by the DCID of the incoming packet.
  - The MPRT is indexed by the DSID of the incoming packet, if present and valid.
  - Each port of a switching component has its own LPRT (and MPRT if appropriate)
  - Refer to Section 8.22 Component Switch Structure, and Section 8.23, Route Control Structure of the Core Spec for details on how to program these two tables to create pathways from an ingress port to an egress port.
  - There are control bits defined by the Core Specification (see Section 8.23 Route Control Structure of Core Specification) which enable more sophisticated look up algorithms to be used between the LPRT and MPRT.
- **Interface Ingress (Egress) Access Key Masks**
  - These Access Key Masks can be used to filter packets based on the A-Key included in the packet, both at the ingress port and the egress port.
  - Only packets with an authorized A-Key are allowed in to or out from a given port if A-Key filtering is enabled on a switching component.
  - See Section 8.16, Interface Structure for details.

- **Link filter controls**: As described later under Domain Boundary Security. (Section 8.16)
  - Ingress DR Enable
  - Control OpClass Packet Filtering Enable
  - Unreliable Control Write MSG Packet Filtering Enable
  - Source CID Packet Validation Enable
  - Peer Nonce Validation Enable

Setting the proper access controls at the Responder involves the following structures:

- **SSAP, MSAP, and Peer-ATTR**, which are the same tables for both Requestors and Responders
  - Responders use the **ACRSP** field when validating permissions of a given Requestor to issue the received packet
- **Responder zMMU R-Key structures**
  - Responder zMMUs provide the R-Key that must match the R-Key in the request packet
- **C-Access structure** (for control space)
  - This structure contains any required R-Keys values for those regions of Control Space which may be accessed by non-manager entities (whose global Source CID does not match the expected PMCID | PFMCID |SFMCID)
  - A non-manager component may be given access to a specific substructure of control space for a given Responder

- For example: performance monitoring node may be given Read-Only R-Keys to the performance counter structures in the control space of numerous Gen-Z switches.

The Union of enabled R-Key Domains and Destinations

Table 2-2: Packet Fields Important to Isolation and Partitioning describes the packet fields that are filled in by values which come from application or OS software and the packet fields that are filled in by the Gen-Z hardware. The table also describes how the Gen-Z hardware (Requestor or Responder) validates the fields of the packet before transmission by the Requestor or after receipt by the Responder.

Figure 2-2 *Important Control Structures Dictate Requestor Packet Contents* indicates that the R-Key chosen for the packet and the destination component (DCID) chosen for the packet both come from software controlled structures which are associated by software with a given access to a logical resource. The permissions are extracted from Gen-Z logic blocks under the control of the Fabric Manager.

Note that though the Application or OS software may choose the 32-bit R-Key value, the Requestor hardware may not be enabled to send packets with that software-proposed R-Key Domain (RKD) value. The allowed R-Key Domain values that can be used are determined by the Gen-Z manager, based upon which connections to which sub-ranges of which Responders this particular Requestor is authorized to access.

However, there is no cross-check by the Gen-Z component that software has paired the *correct R-Key* with the *correct region* (z-address) of the *correct destina*tion (DCID). Figure 2-3: *The Range of Accessible Components and R-Key Domains* offers an example how software has control over the *union* of enabled destination components and the enabled set of R-Key Domains.

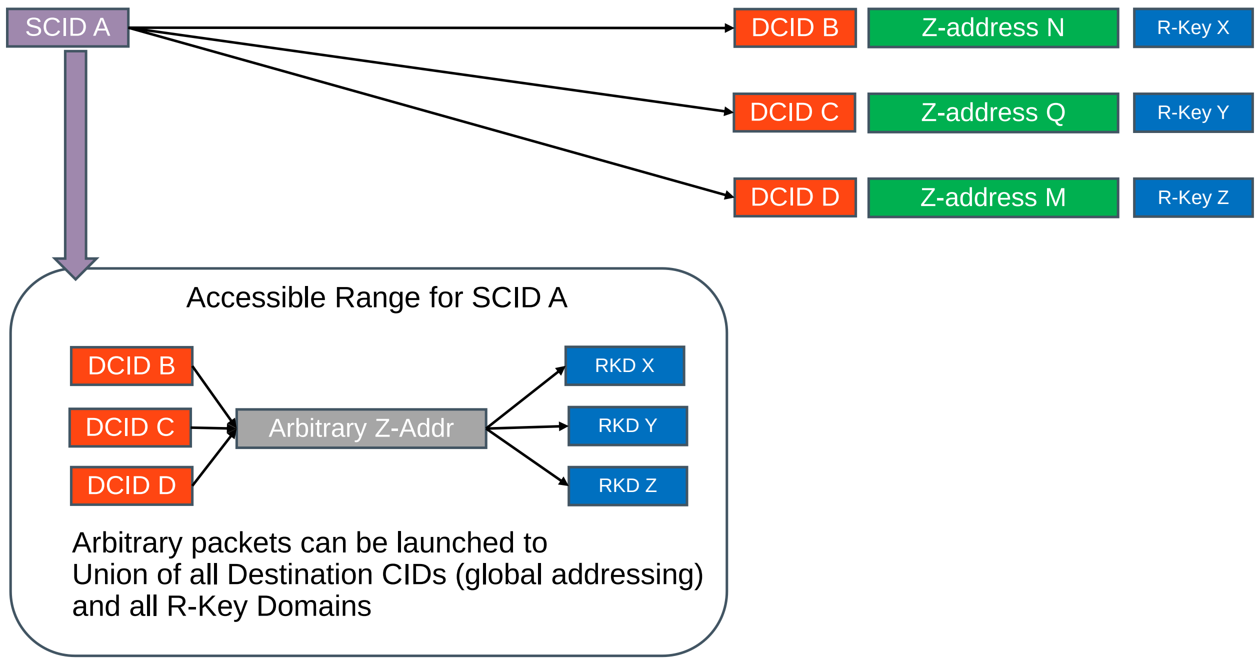


Figure 2-3: The Range of Accessible Components and R-Key Domains

A requestor (SCID A) has been enabled for these specific accesses:

- Destination CID B at Z-address N which requires R-Key X (in R-Key Domain X)
- Destination CID C at Z-address Q which requires R-Key Y
- Destination CID D at Z-address M which requires R-Key Z

However, software can maliciously or erroneously try any combination of the enabled destinations with any enabled R-Key Domain at any z-address, as shown. This means that if destination D (DCID D) also has a sub-region associated with R-Key Domain X (perhaps for the use of a different Requestor, say SCID W), software on Source CID A can launch probing reads to DCID D in hopes of stumbling upon a valid 32-bit R-key that will give SCID A access to data intended only for SCID W. Thus, the Gen-Z management stack is responsible for managing R-Keys and R-Key Domains in a manner which guarantees such un-intended overlaps in accessible ranges cannot occur. How to avoid such overlaps is beyond the scope of this document.

Domain Boundary Security:

Before any component can be enabled to do packet routing or execution, the manager shall determine the peer component's manager domain status for every available ingress link. The necessary details about boundary security are found in Section 4.4 Securing Boundary Links.

2.2.9. Power Management

The Gen-Z management stack is being architected to enable a global view of and management of all resources present on the fabric. A major contribution to the cost efficiency and usability of the aggregate resources is a global power management framework.

The Gen-Z Core Specification (Section 8.9.2) breaks power management features for Gen-Z components into the following classes:

- Physical Layer Power Management
    - Multiple Link / PHY hardware power states are defined and with them are associated multiple Link-CTL packets that inform the connected peer components of pending or completed link power state transitions. See Section 8.16.10 of Interface Structure and Section 8.17 Interface PHY Structure in the Core Specification for additional details.
    - Link and PHY power states can be manipulated by software to explicitly optimize link power utilization under static or slow dynamic changes in bandwidth or latency demands. To optimize these gross-level choices, software power manager codes should reside at a point in the manager software hierarchy where there is visibility to which applications are running on which nodes and which other fabric attached resources are required by those applications.
    - Link and PHY power states can also be coordinated by the components themselves in response to corresponding changes to the component level power utilization changes.

  Since Link and PHY power states are so tightly coupled to the associated component level power state or global fabric utilization choices, our management architecture will not directly address the use of Link Power states.

- Component and Module Power Management
    - Much like link power states, the component level power consumption can be adjusted via **Software-directed Power Management** of a component's C-State for gross-level control. See Section 8.9.2.1 of the Core Specification.
    - To respond to brief, but significantly numerous periods of light or non-existent activity between and within many Gen-Z components, the Core Spec has defined **Automatic (Autonomous) Power Management** of a component's C-State for more fine grained power control with more rapid transitions and shorter durations than Software-directed management. See Sections 8.9.2 and 8.9.3 of the Core Specification for a brief description of how a component might be enabled to adjust its own power consumption based upon programmable idle time constraints and the rules of C-State transitions.
        - Automatic Power Management is controlled by the Automatic C-State Support bit in the Core Structure Component CAP 1 field, and the corresponding Automatic C-State Enable bit in the Core Structure Component CAP 1 Control field.
    -

The rest of this discussion assumes the following context about C-States:

- The component is in one of the operational power states C-UP, C-LP (C-low power), or C-DLP (C-deep low power).
- The power consumption hierarchy is
    - C-UP is the most capable state and is enabled to consume power up to the maximum power level the component advertises in the Core Structure or the Component Mechanical structure.
    - C-LP is the next most capable C-State. Components must process Control class OpClass packets in C-LP. Components must transition to C-UP to handle any other End to End OpClass packets.
    - C-DLP is the lowest power C-State. Components do not process any end-to-end packets, but will acknowledge Link-CTL packets on at least one link.

  See Section 8.4 Component States and Descriptions in the Core Specification for more details.

- Components may transition to a lower power C-State from any higher power C-State
    - Eg, C-Up to either C-LP or C-DLP
    - Eg, C-LP to C-DLP
- Components may only transition to C-UP from either lower power C-State
    - Eg, C-LP to C-UP
    - Eg, C-DLP to C-UP
    - But never C-DLP to C-LP

Software-directed Power Management

Software can request a component to transition to one of the three operational C-State power levels using either of two mechanisms:

- Manager Software with Primary or Fabric Manager authorization can issue Control Space writes to set one of following 'transition control bits' (if supported and enabled) in the Core C-Control fields
    - Transition C-Up (from C-LP or C-DLP)
    - Transition C-LP (from C-UP to C-LP)

  - Transition C-DLP (from C-UP or C-LP)

  Transitions requested via writes to these control bits are not negotiable if the component supports this transition via these control bits. (There are corresponding 'transition capable bits' in the Core Structure Component CAP 1 fields.)

  The component does not return the corresponding Control Standalone Acknowledgement until the component has completed the requested transition, or has failed in its attempt to do so.

- Software with access to a component's Control Space by virtue of being a Primary or Fabric Manager or a Power Manager may issue a Control OpClass **C-State Power Control packet** to the component. See Gen-Z Core Specification (Section 6.10.2.4).
  - These C-State Power Control packets can be used to request a C-State transition,
  - These C-State Power Control packets can also can be used to set a limit (power cap) on the peak power consumed by the component, as a percentage of the maximum power claimed.
  - The source component (the Software host's bridge) and the target component must both be enabled to issue and execute these C-State Power Control packets as 'transition requests'.
  - The C-State Power Control packet can also be used to 'notify' peers that a component is proposing to transition.
    i. Notification functionality must also be enabled for the source and target components before such packets can be issued or executed.

  Software shall not issue C-State Power Control packets as Transition Notifications unless software can guarantee its issuing bridge will respond properly to any 'abort transition' responses received.

  - Normally, a software thread will not have visibility to the NACK reason code and therefore will not have the ability to abort its transition.

A component's C-State is determined by several factors, but manipulated in just a few ways.

Architected Power Management States are defined in the Core Specification. The Fabric Manager will make explicit changes to power settings in the Control Structures of components, but the recommended changes are typically decided by upper level managers such as Resource Managers, Power Managers, or the Composability Manager.

The following are requirements, policies, and assumptions of the Gen-Z management architecture related to power management:

- hardware shall automatically (autonomously) adjust power states only when software enables such features
- Software can program the constraints and enable or disable the automatic (autonomous) power controls
- Software shall not enable automatic power management on components prohibited from transitioning to C-UP from lower power states.

    - This prevents unintended power consumption spikes from exceeding power supply capacity.
    - This is not stated in the core spec, but we've been asked to supply some clarifying text. If the above requirement is adopted by the core spec, this mention will become worded as a reminder.
- A component shall never refuse to transition to C-UP from any lower power C-State.
- Component power (HPWR, NPWR, etc) values reflect actual component power, not anything more.
- Component Power cap limits (% of Max component power) apply only to the component, not anything more.
- Media power fields (Primary Max Media Power) apply only to the media associated with the component.
- How Component and Media power values are determined and how relationships between them are manifest are outside the scope of both the core spec and this document.
- There is no architected method to reveal additional power consumption peaks and control surfaces of any non-Gen-Z components on a given FRU or subassembly.
    - We encourage endpoint vendors and platform vendors to work with DMTF.org to enable MCTP messages/actions/commands which address FRU inventory and power management.
- C-state Power Control packets which 'request' a component to transition may originate with any valid component manager or the component's PWR_MGR
    - the component is expected to comply if so enabled, unless vetoed by peer components when they respond to the advisory C-State Power Control packets sent to them
        - The component, if enabled to do so will signal other components using the C-State Power Control packets in advisory mode
    - Primary or Fabric Manager may issue direct writes to Core Control structure, or may issue the C-State Power Control packets
        - The latter power management packet can set a power cap at a percentage of the maximum in addition to a transition to a specific power state
        - The power cap percentage always refers to the maximum power consumption claimed for the C-UP state.
- When a component issues 'notification' C-State Power Control Packets to peers:
    - If any peer fails to respond, the notification fails and the transition shall not be attempted
    - If any peer responds with the abort transition reason code, then the notification fails and the desired transition shall not be attempted
    - If any peer responds with the abort transition reason code and requests a transition to a different C-State, the component shall transition to the highest power state of those requested in the abort transition acknowledgements.

  - The core spec does not specify how a component knows to which peers it shall send such notifications.

2.3. Gen-Z Management Basics

2.3.1. Assumptions and Requirements

The Gen-Z management architecture assumes the following interpretations and requirements:

- Gen-Z components that bridge a processing element onto the Gen-Z fabric, while often called 'bridges, HBAs, HCAs, Fabric Adapters, etc', are defined in the Core Specification as a base class type of 'Processor' or 'Bridge', eg.:
  1. Processor (Bootable)
  2. Processor (non-boot)
  3. Integrated Gen-Z Bridge
  4. External Gen-Z Bridge

- If a Gen-Z component (e.g. Processor or Bridge) is to perform as a manager entity (processor thread executing a Gen-Z manager role) in any topology which has or potentially has more than one manager entity which negotiates access to shared Gen-Z resources, or is intended to interface a node's Local Node Services to a remote Gen-Z Fabric Manager, that component shall support the following features and capabilities:
  1. The unreliable Control Write MSG opcode and packet format and processing as sender, receiver, or relaying component
     - Support of the unreliable Control Write MSG opcode is optional in the Core Specification, but for these components this is a mandatory feature to enable Manager to Manager communications
     - Including the proper assertions and decoding of the DR and SDR bits within such packets per Section 12.9 of the Core Spec.

  2. A maximum message size of at least 1536 bytes when using any variant of Control Write MSG packets, including the unreliable, directed relay version.
     - This requires these components to support a minimum UCWMSZ such that the product UCWMSZ * Max Packet Payload) is greater than or equal to 1536 Bytes.
     - UCWMSZ and Max Packet Payload are defined in the Gen-Z Core Specification in Section 8.15, in the Core Structure and Core CAP 1 Structure, respectively.
     - For example, if Max Packet Payload value is 0x0 (256 Bytes), then UCWMSZ must be at least 0x6, in which case UCWMSZ * Max Packet Payload = 1536.

  3. The proper manipulation of packet contents for all stages of processing of the directed relay version of unreliable Control Write MSG packets, summarized in 4.2.1 *Initiating the Manager to Manager Communication*.
  4. A default Responder Context ID (RSPCTXID) == 0 reserved for management use of Control Write MSG packets

        - Gen-Z management software is not restricted to *only* RSPCTXID == 0, but that is the only RSPCTXID that is guaranteed to be valid with Control Write MSG

    5. A Completion Handler as defined for the Control Write MSG and Responder Context ID 0
        - EG. A component / platform dependent method (GSE, interrupts, SMI) of invoking a Completion Handler ISR in the manager role context of the associated processor after the Control Write MSG has be deposited into the buffer.
        - It shall be possible to configure the receiving bridge to invoke the Completion Handler as an OS or application thread, or as a System Firmware thread.

    6. The generation and receipt of Unexpected Event Packets.
    7. The Control Read and Control Write opcodes of the Control OpClass, including the directed relay versions.
    8. At least one component local (platform specific) interrupt which can be signaled to system software or firmware running on the host processor.
        - The Core Spec does not require that any of the possible platform specific interrupts which can be triggered by the bridge must be delivered to a host thread. However, to support manager code to run as a system firmware or OS/app thread on the host, we require this ability.
        - Further, if only one host context can be signaled via such an interrupt, it shall be the system firmware context so that 'firmware first' event handling strategies can be applied.
        - It is highly recommended that bridge vendors supply at least two platform specific interrupts, and that platform vendors deliver at least two such interrupts to host threads: one of which can be dedicated to delivery to the SFW context and one of which can be delivered to the OS /application context.
    9. An appropriate set of Flush and Fence controls, as outlined in *2.3.5 Flushing and Fencing End-to-End Requests.*

- Gen-Z endpoints, switches, transparent routers, and manager entities all have platform specific out of band initialization capabilities (HW Prep) that set up the Gen-Z components appropriately, leaving them enabled for a compatible in-band management entity to come in through specifically enabled link interfaces and take control of the in-band management structures.
    1. There will be some components that will need certain core control structure values be set to non-default values via this HW Prep phase depending upon their intended use within a greater Gen-Z environment.
    2. Out-of-band mechanisms may be used to establish additional 'non-default' initial values on components.
        - EG: The 'manager type' bit defaults to 0b (type == Primary Manager) per the Core Specification. However, fabric switches and many types of Gen-Z components destined to be shared are expected to exit HW Prep with the 'manager type' bit set to 1b (type == Fabric Manager), so a Fabric Manager will attempt to take control of such a component immediately.
- The networking and messaging connectivity are not necessarily the same as the FAM, Compute, and Storage connectivity.
    1. Connections are always between a Gen-Z Requestor and a Gen-Z Responder

   2. One or more 'paths' or 'routes' exist between connected Requestors and Responders
   3. Not all possible paths or routes between specific Requestors and Responders will always be enabled
   4. Networking or Messaging Resource Managers may have entirely different sets of desired and prohibited routes when compared to the routes desired by a FAM Resource Manager
   5. The Route and Topology Manager or other clients of the Composability Manager owns resolving direct conflicts concerning allowable routes between a specific Requestor and Responder

- Networking, messaging, FAM, Compute, and Storage resources may be controlled by different resource managers; all such resource managers call into the Gen-Z Primary Manager or Primary/Secondary Fabric Manager which proxies the actual hardware changes.
- All manager entities allowed to control Gen-Z components that attach to a given fabric are trusted to the extent any firmware updates to the manager entities are trusted.
- All manager entities in all roles that can request or directly make control space changes to Gen-Z components' Core Control structures are fully trusted and authenticated. See *section 4, Dealing with Multiple Managers.*
- All manager entities shall have a MGR-UUID (manager UUID) that identifies the code instance as unique, or as a collective of code instances all of which are executing as a common, distributed instance.
   1. Each instance of a manager entity shall have a means of generating a MGR-UUID which is globally unique from any other instance.
- Control Space reads and writes are not idempotent, and therefore can arrive multiple times in any order at the Responder. All manager read or write sequences to control space shall account for re-ordered or repeated control space reads and writes to be received at the Responder.
- The core specification defines a clear priority among the possible multiple managers of a component based on the setting of the 'manager type' bit in the control space core structure.
   - Setting the 'manager type' bit to 1b == 'Fabric Manager' will cause control space packets issued from the Primary Manager (PMCID) to be ignored.
   - Setting the 'manager type' bit to 0b == 'Primary Manager' will cause control space packets issued from a Primary or Secondary Fabric Manager (PFMCID or SFMCID) to be ignored.
- It is recommended that component control space be mapped to Requestor pages with the 'processor exception control' bit set to '1b' whenever that capability is supported by the processor bridge component. Silently dropped reads of control space will thus return all 1s to the host rather than generate a memory reference fault.

- The first write to an un-managed component's control space should be the CID0 control field and, if appropriate, the SID0 value. This gives the component a local subnet address, which enables the manager entity to directly address the component and eventually eliminate the need for 'directed relay packets'.
- *Un-managed components*, while in C-CFG state, will capture the SCID (& SSID if valid) of the *first* control space *write* it executes and copy it into the PMCID or PFMCID control field depending upon the state of the component's Manager Type bit and the control write packet's GC bit, per Section 8.6.1 of the Core Spec.

- A managed component will have a valid PMCID field if the manager type bit == 0b, or a valid PFMCID if the manager type bit == 1b. A managed component will not yet have a valid CID (and optional SID) if the owning manager has not yet assigned one.
    1. The manager can only communicate with a component that has no valid CID via directed relay packets.
- Components shall not be enabled to C-UP if they do not have a valid CID 0 assigned.
- Components can only enter C-CFG state from the C-DN state and thus the component must be sent through a cold or warm reset to reach C-CFG.
- Gen-Z components transition from power-off to a state where they are ready to be in-band managed in phases:
    1. **Reset (warm or cold)**
        - Hardware exits reset as described in Section 12 of the Gen-Z Core Specification
    2. **HWInit**
        - Hardware dependent values get produced and locked down into Control Space structures in vendor specific manners
        - Hardware exits the HWInit phase when the Core C-Status HWInit Valid bit transitions from ‘0b’ to ‘1b’.
    3. **HW Prep**
        - Some additional platform specific settings get installed and leave the component ready to auto-train links and be in-band managed (if enabled)
        - HW Prep phase shall be complete before or concurrent with when the component is enabled for in-band management packet processing and at least one Gen-Z link becomes enabled to auto-train and the associated interface is enabled to enter I-CFG.

2.3.2. Domains, Namespaces, and the Grand Plan

The Grand Plan

- There should be a grand plan for the run time state of the entire Gen-Z fabric.
- This grand plan defines an overall Gen-Z management scheme, be it one manager in a 'flat' management control plane or multiple, cooperating managers overseeing the fabric in a segmented or hierarchical fashion.
- This grand plan defines how each manager entity creates its unique MGR-UUID for use in multiple manager topologies.
- This grand plan defines the policies for creating a complete Gen-Z CID namespace for all the subnets and all the resources, and defines the policies on how to handle dynamic configuration changes.
- There is thus a 'local' grand plan for each of the Gen-Z Primary and Fabric Managers wherein their domains of control, their roles and responsibilities, and their relationships and priorities with respect to each other are defined.
- This grand plan includes desired end states and transition policies for initial setup, restart, and dynamic changes of the entire fabric and its CID namespace.
- The important details about the grand plan are made available to the appropriate manager entities before such entities complete the initial setup, restart, or dynamic changes of the fabric.
- Primary Managers either have a priori information about the role of their resources in the grand plan, or they follow a prescribed set of policies using architected mechanisms to learn of and participate in the grand plan from a Fabric Manager.
- The grand plan defines how fabric and Primary Managers will interface to the higher level resource managers which have the final say on which connections and paths are to be enabled and how they are to be protected.
- When situations are encountered where the grand plan does not provide clear policies for the managers to proceed, default discovery and enumeration policies should be applied so as to secure the proper, safe operation of those portions of the fabric that do have clearly defined roles.
- In the absence of any a priori 'grand plan', every manager instance shall follow the default set of Rules of Exploration defined in 4.1 *General Manager Domain Exploration Policies*.
- All managers should have a default interface through which an admin can request additional setup actions or can supply information missing from any a priori grand plan.
- No manager shall alter persistent data contents at any time without clear guidance from a grand plan or an administrative authority.

2.3.3. Important Control and Status fields in the Control Structures and Their Role in Configuration

2.3.3.1. Ownership Controls

The following table is a collection of the status and control bits listed in the Gen-Z Core Specification revision 1.1 document which are used to determine how ownership of management privileges are established. This list is made from the perspective of a single subnet. Thus, there are few references to the SID and global CID group controls and IDs. Extension of discussion to global IDs is left to the reader at this time.

Table 2-3: Important Ownership Controls and their Purpose

CSR field name	Value (Boolean listed as 0 / 1)	purpose	location	1.1 reference	Interpretation of impact
CID	12 bit ID value	Component CID	Core Structure	Table 8.3	Set by in-band or out-of-band manager, or platform dependent mechanism Up to 4 CIDs can be assigned to a component, but there is only one control space, one data space., and one Component Subnet ID (Component SID).
PMCID	12 bit ID value	Declares the CID of the Primary Manager	Core Structure	Table 8.3	Set by current manager or by HW via capture
PFMCID	12 bit ID value	Declares the CID of the Primary Fabric Manager	Core Structure	Table 8.3	Set by current manager or by HW via capture (see section 8.6 of core spec)
SFMCID	12 bit ID value	Declares the CID of the Secondary Fabric Manager	Core Structure	Table 8.3	Set by current manager (not set by capture)
MGR-UUID	128-bit UUID	If non-zero, this is the UUID of the one, true manager intelligence	Core Structure	Table 8.3	If this UUID field is non-zero, any control packet from any of the in-band managers MUST include a matching MGR-UUID or the packet fails for Access Permission.

FRU-UUID	128-bit UUID (optional)	If non-zero, this is the UUID assigned to all the Gen-Z components found on this instance of a FRU	Core Structure	Table 8-3	If available, allows Gen-Z manager to discern which components are part of the same physical Field Replaceable Unit
Component Enable	Boolean Disabled / enabled	Disabled == "C-down", enabled == "C-Up"	Core C-Control	Table 8.5	Component overall "up or down" state will follow the state of this control bit. Low power states (C-LP, C-DLP) are sub-states of the 'up' or 'component enabled' state. C-CFG is a transient state between C-DOWN and C-UP.
Manager Type	Boolean: Primary Manager / Fabric Manager	Indicates which manager ID (PMCID or P/SFMID) is used to validate control space packets	Core Capabilities 1 Control	Table 8.8	Setting this bit to '1b' will disable validation against the PMCID field. Only packets whose SCID matches the PFMCID or the SFMCID are validated.
In-band Management Enable	Boolean: enabled / disabled	Indicates if in-band mgmt is enabled	Core Capabilities 1 Control	Table 8.8	Establishes and indicates if component responds to any in-band control packets
Out-of-Band Management Enabled	Boolean: enabled / disabled	Indicates if out-of-band mgmt is enabled	Core Capabilities 1 Control	Table 8.8	Establishes and indicates if component responds to out-of-band control requests
Primary Manager Role	Boolean Not here / here Sticky across soft resets –	Indicates the primary manager is co-located at this CID	Core Capabilities 1 Control	Table 8.8	The local CID and its valid bit are copied into the PMCID control fields before this bit is set. While this bit is set, the PMCID control fields are 'read-only', unless the Primary Manager Transition Enable bit is set. Use the sticky attribute of this bit to indicate intended manager type upon warm reset.

Primary Fabric Manager Role	Boolean Not here / here	Indicates the primary Fabric Manager role is co-located at this CID	Core Capabilities 1 Control	Table 8.8	The local CID and its valid bit are copied into the PFMCID control fields before this bit is set. While this bit is set, the PFMCID control fields are 'read-only' unless the Fabric Manager Transition Enable bit is set.
Secondary Fabric Manager Role	Boolean Not here / here	Indicates if secondary Fabric Manager role is co-located at this CID	Core Capabilities 1 Control	Table 8.8	The local CID and its valid bit are copied into the SFMCID control fields before this bit is set. While this bit is set, the SFMCID control fields are 'read-only' unless the Fabric Manager Transition Enable bit is set.
Primary Manager Transition Enable	Boolean Disabled / enabled	Enables component to accept new CIDs into PMCID while Primary Manager Role bit is set	Core Capabilities 1 Control	Table 8.8	Over rides the 'read-only PMCID' rule when Primary Fabric Manager Role bit is set. Enables a co-located PM to give away its role by writing a new PMCID value.
Fabric Manager Transition Enable	Boolean Disabled / enabled	Enables component to accept new CIDs into P/SFMCID	Core Capabilities 1 Control	Table 8.8	Over rides the 'read-only PFMCID' rule when Primary Fabric Manager Role bit is set.
Software defined sticky bits	8 bit field, sticky across soft resets	8 sticky bits (RWS) for software use	Core Capabilities 1 Control	Table 8.8	Software (OOB or in-band) can set these bits as needed. Bits retain their state across soft (warm) resets. Cleared by hard (power on) reset. Use these bits to pass SW state information across warm resets. (See Section 7

2.3.3.2. Important Link Control Fields

The following table is a collection of the status and control bits listed in the Gen-Z Core Specification revision 1.1 which determine which types of packets may be accepted by a given link interface and relayed to another port or a Responder on the component. This list is made from the perspective of a single subnet. Thus, there are few references to the SID and global CID group controls and IDs. Extension of discussion to global IDs is left to the reader at this point.

Table 2-4: Important Link Control Fields Used for Discovery and Configuration

field name	Value (Boolean listed as 0 / 1)	purpose	location	1.1 reference	Interpretation of impact
Ingress DR Enable	Boolean Disabled/ Enabled	Enable directed relay packets at this component	Interface I-Control	Table 8-25	Set this enable true to allow control space packets with the DR bit ==1 to ingress at this port and execute on this component.
Directed Relay Control Write MSG Packet Filtering Enable	Boolean Disabled/ Enabled	Select for directed relay Control Write MSG packets	Interface CAP-1 Control	Table 8-27	Set this enable true to further limit the types of DR packets allowed in through this link to just the Control Write MSG opcode.
LPRT Enable	Boolean Disabled/ Enabled	Master enable for ingress port route lookups using the LPRT	Interface CAP 1 Control	Table 8-27	Setting the LPRT Enable false will cause all LPRT look ups to fail for reason of 'Switch Packet Relay Failure'
MPRT Enable	Boolean Disabled/ Enabled	Master enable for ingress port route lookups using the MPRT	Interface CAP 1 Control	Table 8-27	Setting the MPRT Enable false will cause all MPRT look ups to fail for reason of 'Switch Packet Relay Failure'
Control OpClass Packet Filter Enable	Boolean Disabled/ Enabled	select for Control OpClass packets	Interface CAP 1 Control	Table 8-27	Setting this filter enable true will cause all non-Control OpClass packets to be rejected.

initiate Peer-C-Reset	Write only	Write 1 to send C-Reset to Peer	Interface I-Control	Table 8-25	Use to reset component on the other end of this link (if feature is enabled)
Link RFC Packet Disable	Boolean Enabled / Disabled	Enable link to send RFC Packets	Interface I-Control	Table 8-25	Set this bit to disable link from sending RFC Packets when link resets and comes back up
OpClass Select	3-bit control field	Specifies which, if any OpClass packets this interface is enabled to exchange with the Peer Component	Interface CAP 1 Control	Table 8-25	Management must set this field before the interface will exchange anything other than Link CTL packets with the Peer Component on the other end. Management may set this field directly with a Control Write packet. This field may also be set when a valid Peer-Set-Attribute LinkCTL packet is received from the other end.

2.3.4. Auto-training links and Link Level Control Packets

This section describes some important Gen-Z link and interface features and functions and how they are used in the Gen-Z in-band management architecture.

Link Auto-training Requirements

The Core Specification dictates that in-band managed components must eventually reach a state where at least one link will auto-train when connected to a Gen-Z peer component. Further, upon a link training in this manner, the component will begin issuing Link CTL Request for Configuration packets to alert an in-band manager that there is a new component in need of configuration. Upon issuing a Link CTL RFC packet on at least one interface, the new component will transition to C-CFG and await control space packets from a manager.

However, the Core Specification does not define how the link PHY is actually configured with the correct parametric settings to effectively auto-train when a peer PHY is detected. Much of these details are spelled out in the Gen-Z PHY specifications found at https://genzconsortium.org/specification/gen-z-core-specification-1-1.

- All components shall have a component-vendor defined mechanism that will enable platform vendors to configure the following link attributes of the component by the end of the HW Prep phase:

  - Select one or more component links to auto-train (see following requirements and policies)
  - Select the proper PHY type for each link or the whole component (see following)
  - Select the in-band manager type for the component
  - Select the proper PHY reach and speed, as appropriate per below
  - Select the proper PCIe PHY port mode, as appropriate per below

Developer's Note: The methods used by the component vendors to obtain the platform specific configuration values from the platform vendors are not defined in this document or in the core specification.

Choosing PHY Initial Configurations

In the case of component links that use multiple types of PHYs or multi-speed PHYs, there may be platform specific PHY configuration choices that must be made as part of the HW Prep phase of such components. For example, it may be necessary to choose among options such as Local vs Fabric reach loss budgets on 802.3 PHY or Downstream vs Upstream port orientations on PCIe PHYs.

To enable successful plug and play behavior on Gen-Z components using the PCIe PHY, the following policies apply:

- SoCs, Processors, host Bridges (typically claiming to be Base Class of 'Processor' and hosting major Requestor functionality that will originate request packets) shall initially assume PCIe downstream port (PCIe-DP) parameters
  - This enables Primary and Fabric Manager nodes to initiate control space requests
  - Should manager threads detect that PCIe-DP links are trying and failing to train (as would be the case if two host bridges were directly connected together), a vendor defined software policy may be invoked to *randomly* try the opposite port configuration until the link trains
    - Similar to random back-off arbitration tactics for collisions
- Media controllers, e-NICs, and other components typically acting as Responders shall initially assume PCIe upstream port (PCIe-UP) parameters
  - This enables these components to accept in-band management packets from the manager nodes
- Switches (enclosure or fabric class), if using the PCIe PHY, shall initially assume PCIe-UP parameters
  - This enables switches to respond to in-band management packets from in-band managers
- Once an in-band manager takes control of a switch or other component with multiple PCIe PHYs in PCIe-UP mode which are failing to train, the manager may try randomly selecting the opposite mode.
  - Putting un-trained switch ports into PCIe-DP mode will enable media controllers, e-NICs, and other switches still using PCIe-UP mode to successfully connect to the fabric.
- Un-managed components shall not randomly switch modes.
  - Un-managed components wait until a manager takes control of the peer component at the other end of the link and then swap the peer component's mode.
  - In general, un-managed components without a co-located manager will start in PCIe-UP mode and wait for a manager to find it.

To enable successful plug and play behavior on Gen-Z components using multi-speed or multi-reach 802.3 PHYs

- Multi-speed PHYs shall start at lowest speed first unless HW Prep overrides that default.
    - After links initially train and the components are under manager control, the manager may enable link speed upshifting if both components are capable.
- Multi-reach PHYs shall start at fabric settings first unless HW Prep overrides that default.
    - After links train, and the components are under manager control, the manager may use in-band commands to optimize the reach under software control.
    - After links initially train and the components are under manager control, the manager may enable hardware link reach optimizations if both components are capable.
        - At this time, there is no hardware link reach optimization architected. Any such architecture is beyond the scope of this document, and would be documented in the Gen-Z PHY specs.

Components may offer multiple types of PHYs. EG, an enclosure switch may support both PCIe PHY and 802.3 PHY on a per link or per component basis. Such flexibility would allow the enclosure switch to interface with SoC or CPU sourced PCIe PHY links, and use 802.3 PHY uplinks to external Fabric switches. Unfortunately, the 802.3 PHY is incompatible with the PCIe PHY and the two can never train a joint link.

- Components that offer multiple types of PHY which are incompatible with the wrong types shall have an out-of-band mechanism with which the platform vendor can set the choice as part of the HW Prep phase of in-band bring up.

Link CTL Packets and Link RFC Packets: Defined in Section 11.11 of the Core Specification, Link CTL packets are used to transfer critical data and commands between a managed component and an un-managed component when in-band managers cannot explicitly address the un-managed one. Link RFC (ready for configuration) packets are defined in Section 11.6. The following paragraphs discuss types or functions of Link CTL and Link RFC packets that this in-band management architecture relies upon and how they are used for basic device discovery, enumeration, and configuration.

- Peer-Attribute Link CTL packet
    - Link hardware exchanges these packets to query the peer component on the other end of the Gen-Z link
    - Gen-Z management reads the resulting data from the near end link interface control structure as part of the initial discovery process
- Peer-Set-Attribute Link CTL packet
    - Once management determines that a link connecting a managed component to a peer component needs to be enabled for end to end packet flow, management triggers the local link interface to exchange a Peer-Set-Attribute Link CTL packet to set the OpClass and Link-level-reliability (LLR) modes to be use on the link.
- Peer-C-Reset Link CTL packet
    - Management may trigger (if link interface is enabled to do so) A Peer-C-Reset Link CTL request packet to be sent to the peer component, where it will cause the component (if so enabled) to send the Link CTL ACK Link CTL response packet and then perform a full component reset.
    - Management should override the default Transmit Peer-C-Reset Enable bit and Receive Peer-C-Reset Enable bit of fabric managed components by setting both these bits to ‘1b’ before bringing components to C-UP. This allows Gen-Z managers to issue full component resets to

any of their components with simple Link CTL packets should the target component suffer a lock up in the Component's Responder and ignore End-to-End Control Writes.

- Link RFC (Ready for Configuration) packet
  - Link RFC packets are generated when in-band managed components have a link transition to the L-UP state. These packets signal the presence of a potentially un-managed component on the fabric. (See Section 11.1 and Table 11.1 of the Gen-Z Core Specification)
  - Management should disable Link RFC packets by setting the corresponding Link RFC Packet Disable bit = '1b' once a component is configured and ready to transition to the C-UP state.
    - Doing so will prevent individual links that get reset while the component remains in a functional state (C-Up, C-LP, C-DLP) from issuing Link RFC packets when the link comes back up.
    - Link RFC packets should not be issued from configured components.

Peer State Bits Accessible at Link Interface

Peer Attribute update commands request the target link controller to return pertinent information about its component to the requesting link controller, which in turn will make the data available to its manager. Thus, the manager in control of one end of a link can determine important details about the unknown component (peer component) on the other end of the link.

Important Peer Attributes

Table 2-5: Important Peer Attributes Used During Discovery and Configuration summarizes some of the important information that a link interface obtains about the Peer Component attached at the other end of a link. The manager of a configured component can determine several useful details about the Peer Component attached to the other end of a link driven by a given interface of the configured component.

These data are particularly helpful during Enumeration and Discovery, because they allow a Gen-Z in-band manager to quickly determine if a Peer Component is already configured and in an operational state (C-UP, C-LP, C-DLP), the type of manager the component has or is looking for, the OpClasses which can be supported by the Peer Component, and the fabric IDs given to the Peer Component (CID / SID).

Table 2-5: Important Peer Attributes Used During Discovery and Configuration

CSR field name	Value (Boolean listed as 0 / 1)	purpose	location	1.1 reference	Interpretation of impact
Peer Component Manager Type	Boolean Primary Manager / Fabric Manager	Reflects state of Peer component management type bit	Interface Structure Fields – Peer State	Table 8-30	Status bit – This is a copy of the Peer Component's Manager Type bit
Peer Base C-Class	16 bit base class value	Base class value of component on other end of this link, if valid (valid bit in table 8-30)	Interface Structure Fields	Table 8-21	Read this register to find out what type of component is attached to the other end of a Gen-Z link.

Peer CID	12 bit CID value	CID of peer component on other end of this link, if valid (valid bit in table 8-30)	Interface Structure Fields	Table 8-21	Read this register to find the CID of the component attached to other end of a link
Peer Interface ID	12 bit interface number	Indicates the interface number assigned by the Peer Component to this link, if valid (valid bit in table 8-30)	Interface Structure Fields	Table 8.21	Read this value to determine which egress port the Peer Component would use to relay a packet to this link
Peer C-State	3 bit C-State descriptor	Indicates the Peer Component's C-State	Interface Structure Fields – Peer State	Table 8-30	This value can help a manager determine if the Peer Component is already configured.
Peer CID/SID valid	Booleans (one for each field)	Indicates if the Peer CID field is valid	Interface I-Cap1 Control	Table 8-30	These bits validate the Peer CID and Peer SID values
Peer OpClasses Support	Booleans (one for each OpClass type)	Indicates if Peer Component supports each of the following: P2P-Core, P2P-Coherency, P2P-Vendor-defined, and Explicit OpClass	Interface Structure Fields – Peer State	Table 8-30	Use to Discern P2P from Explicit links, or to choose which OpClass to use if more than one type is supported. This information is used by the manager to set the Interface's OpClass Select control field.
Peer In-Band Management Disable	Boolean Enabled/ Disabled	Indicates if in-band management is *Disabled* on peer	Interface Structure Fields – Peer State	Table 8-30	This is a copy of the In-Band Management Disable bit from the peer component's CAP 1 Control structure
Peer Nonce	64-bit volatile value	Manager supplied 64-bit ID. The Peer Nonce Validation Enable is defined Table I-CAP 1 Control.	Interface Structure Fields	Table 8-21	OPTIONAL structure. Used by Manager to ID a specific component; can be checked by HW to validate peer component did not change during C-DLP state

OpClass selection

Section 8.6.1 of the Core Spec indicates that the manager inspects the Peer Attribute fields of the newly operational link to select a compatible OpClass to enable on the link. The management triggers a Peer-Set-Attribute Link CTL packet exchange on the link to enable end to end packets with the selected OpClass. Management involvement is thus required in a default link bring up sequence to choose the OpClass at one end of a link.

Programmers Note: *There is currently no mechanism to pre-assign both ends of the link to the same OpClass and complete the link configuration without manager intervention.* See Link States and Descriptions in Table 11-1 of the Gen-Z Core Specification.

Determining the Manager of a Newly Discovered Component

Developer's Note: One of the first details an exploring manager will need to determine about a newly discovered component is whether it is completely configured, partially configured, or has no manager and is awaiting configuration. Section 8.6 of the Core Specification outlines the general process of how a manager can take control of an unmanaged component, but does not discuss how to first discern the management state of the new component. Unfortunately, for the sake of optimum security, the Peer Attribute fields do not directly define the CID and MGR-UUID of the peer component's manager (if any). However, given the information available in the Peer Attribute fields, it is possible for the exploring manager to make the following inferences:

- If the peer component's C-State is C-UP, C-LP, or C-DLP the component is operational, and therefore managed. See Section *4.2 Manager to Manager Communication* for details on how to initiate communications with a peer component's manager.
- If the peer component's C-State is not operational, but its CID field is valid then the component has a manager that has at least partially configured the component.
    - It is possible that the peer component is actually managed by this exploring manager via another link. To determine if this exploring manager is a valid manager for a managed component, the exploring manager should send a Directed Relay Control Read to the component, potentially reading the manager ID valid bits. If a successful response is received, the peer component does recognize this exploring manager as a valid manager.
    - If no response is returned by the peer component then this exploring manager is not already installed as a valid manager. See Section *4.2.1 Initiating the Manager to Manager Communication* for details on how to initiate communications with the peer component's manager.
- If the peer component's C-State is C-CFG or C-Unknown and its peer CID field is not valid, then the component is *potentially* un-managed. If this exploring manager examines the peer component's Manager Type bit and In-band Management Support bit and determines that it desires to take control of this component, then this exploring manager should send a Directed Relay Control Read to the component, potentially reading the appropriate manager ID valid bits. If a successful response is received, the peer component either has no manager or has this exploring manager as a valid manager.
    - If the appropriate manager CID valid bits are not set, the component has no manager. The exploring manager should attempt to write the new component's CID field to capture control over this component.
    - If the appropriate manager CID field is valid, this exploring manager's CID value must match one of the new component's manager CID fields (PMCID if Manager Type == 0b, or one of PFMCID or SFMCID if Manager Type == 1b) else there is something wrong in the peer component hardware.

2.3.5. Flushing and Fencing End-to-End Requests

The Unicast Request packet Reliable Delivery end-to-end request-acknowledgment paradigm leads to out of order delivery and non-deterministic response times. Only the initiating Requestor is capable of validating that any given Gen-Z request has been received and executed by the intended Responder. Therefore, the Requestor logic is required to support mechanisms to flush its queues and perform fencing functionality.

The following are the features and requirements associated with reliable request packet Flushing and Fencing:

- Reliable request packet exchange requires the Requestor logic track outstanding requests that are awaiting a positive acknowledgement.
    - The mechanisms the Requestor uses to do this tracking are vendor specific.
    - The mechanisms a host uses to trigger Gen-Z transactions through a Requestor are vendor specific. E.g., Requestors may issue Gen-Z unicast OpClass Reads or Writes in response to host load or store instructions, or Requestors may issue Gen-Z unicast OpClass requests such as Reads, Writes, Atomics, or Buffer operations in response to vendor-specific data mover operations. Each of these sources of Gen-Z operations may have their own completion tracking queues in a Requestor.
    - The methods that identify which source generated which packets in the Requestor's tracking queues are vendor dependent.

2.3.5.1. Flushing End-to-End Requests

- Requestors that issue reliable Gen-Z packets shall support a vendor-defined method for flushing and fencing all or a subset of all outstanding requests that are awaiting responses or standalone acknowledgements from the intended Responders.
    - **Flush All**: All Requestors shall support a flush operation that completes only after all outstanding requests to any and all Responders have completed.
        - Components with multiple CIDs shall have a mechanism to indicate if the flush operation is targeting all Requestors of the Component, or just the Requestor associated with a specific CID.
    - **Flush Specific Responder**: Requestors shall support flush and fence operations that complete only when all outstanding requests targeting the Responder associated with a specific CID have completed. Requests that do not target the specified Responder shall not be impacted.
    - **Flush Specific Requestor Source**: Requestors that support generation of explicit OpClass packets from multiple sources, such as load/store host interfaces, data movers and other vendor specific packet generation mechanisms may support flush and fence operations that complete only when all outstanding requests from the specified source (targeting any Responder) have completed.
    - **Flush Specific Connection**: Requestors that support generation of explicit OpClass packets from multiple sources, such as load/store host interfaces, data movers and other vendor specific packet generation mechanisms may support flush and fence operations that complete only when all outstanding requests from the specified source to the specified Responder have completed.
    - **Flush Specific Requestor - Responder Context Pair**: If a Requestor can track outstanding packets issued from a specific Requestor's Context (RESPCTXID) to a specific Responder's

    Context (RSPCTXID), the Requestor may support flush and fence operations that complete only when all outstanding requests made between the given REQCTXID and RSPCTXID have completed.

- The methods utilized to initiate one of the above defined flush operations and signal its completion are vendor defined, but shall include
    - A vendor-defined **Flush/Fence Capability structure** which describes which Flush / Fence operations and features are supported
    - A vendor-defined **Flush/Fence Control structure** which contains fields to indicate the necessary details of the operation being initiated
    - A vendor-defined **Flush/Fence Status structure** which describes the state of any Flush/Fence operation being initiated by the associated Control structure
- A Requestor may support more than one Flush/Fence structure to support multiple Flush operations in parallel
- A Requestor may support programmable Flush / Fence structures to enable software to select specific matching characteristics to use as the flush / fence parameters.
- A Requestor which supports a load/store interface shall support at least one Flush/Fence structure devoted to the load/store interface which is separate from any Flush/Fence structures devoted to hardware data movers, RDMA engines or other asynchronous sources of requests.
    - Host software is responsible for virtualizing a limited number of Flush/Fence structures available to software.
    - Hardware shall map each available Flush/Fence structure to a different page to enable this virtualization

Developer's Note: *Component Developers are encouraged to provide the following Flush/Fence structures:*

- *At least one controlling structure per hardware data mover, RDMA, DMA, or other asynchronous source of requests being injected on the Gen-Z fabric by the Requestor.*
- *At least one controlling structure per hardware thread supported by the host*

2.3.5.2. Fencing Behavior

- Requestors that support any form of Flush operation shall also support an associated Fence behavior that
    - Can be simultaneously triggered when the Flush operation is triggered
    - Stalls any subsequent requests that would also be tracked as a member of the subset of all outstanding requests being flushed
        - The mechanism to stall subsequent requests during a Fencing operation is vendor dependent, but shall include stalling reads of the Flush / Fence Status location until the operation is complete
    - Interlocks with any other Flush or Fence operation that impacts a common source context or destination component.
        - Interlocked Flushes or Fences shall not complete until the union of all affected packets have been processed and acknowledged.
        - All subsequent requests which conflict with any Fence in progress shall be stalled (not issued to the fabric), regardless of source.

3. Example Gen-Z Management Use Cases

In this chapter we will first look at the general flows of both a Gen-Z Primary Manager and a Primary Fabric Manager as they crawl out a Gen-Z fabric, discover and configure components, and then converse with other managers to integrate components into functional systems.

3.1. Gen-Z Management for Moderate Sized Systems

Consider the following moderately sized Gen-Z system which contains a representative large compute node, a remote enclosure full of persistent memory modules, a pair of (redundant) fabric switches, and a small compute node dedicated to running the Gen-Z Fabric Manager entity.

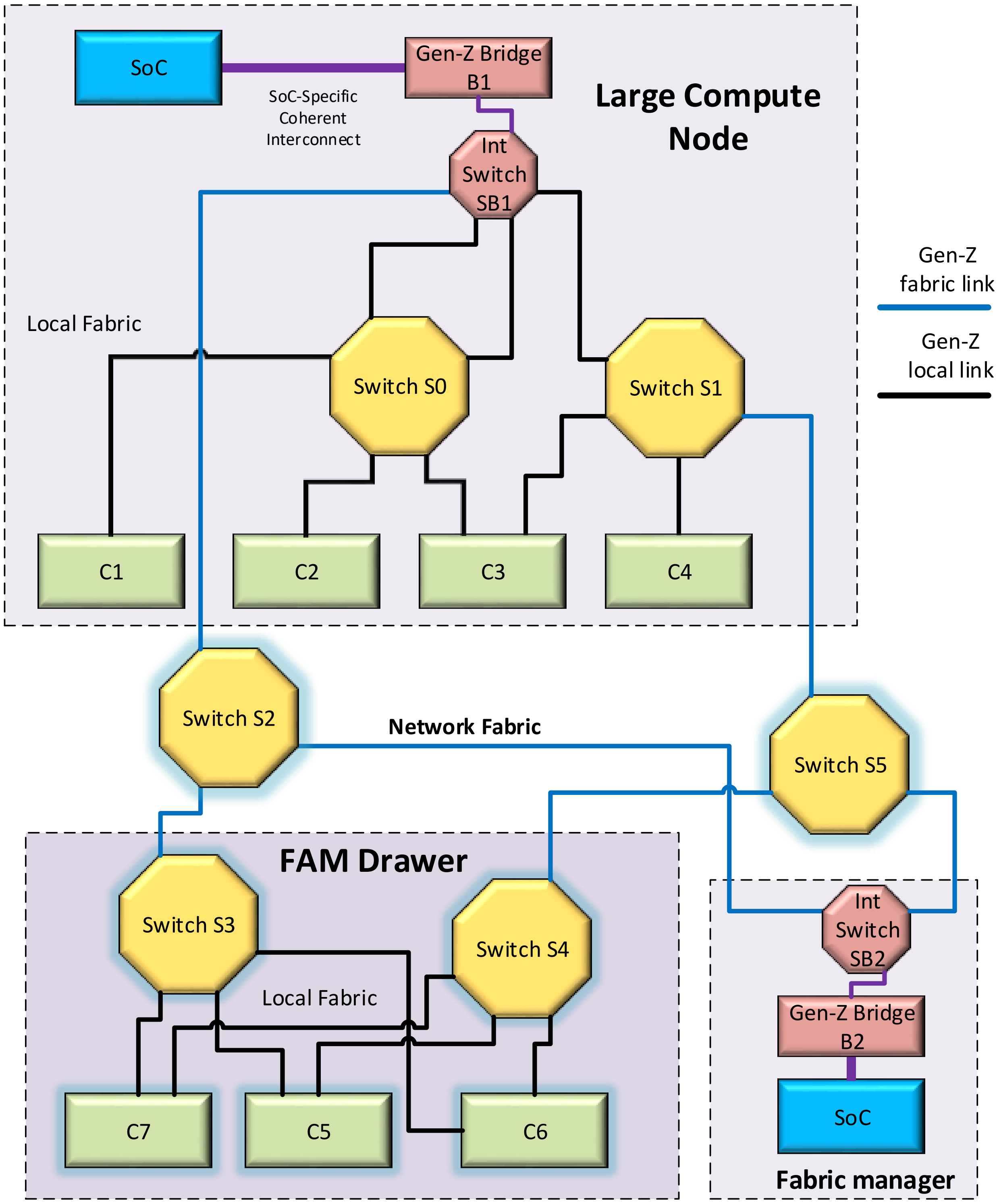


Figure 3-1 Moderate Sized Gen-Z System

Component Key

- Large Compute Node
  - Local Fabric
    - C1, C2, C4: media controller or other endpoints, core64 opClass
    - C3: media controller with redundant paths, core64 opClass
    - S0, S1: Gen-Z Switches
    - B1: Gen-Z bridge (Processor base class with integrated switch SB1)
    - SoC: Any compatible CPU/SoC
  - Gen-Z Network Fabric
    - SB1: one Gen-Z port connected off node, core64 opClass
    - S1: one Gen-Z port connected off node
  - Management
    - Primary Manager for initial crawl out and standalone operation
    - Fabric Manager takes over when node is admitted to fabric (composed)
- Gen-Z Network
  - Gen-Z Network Fabric
    - S2, S5: Gen-Z switches
  - Management
    - Fabric Manager only
- FAM Drawer
  - Local Fabric
    - S3, S4: Gen-Z switches
    - C5, C6, C7 media controllers, core64 OpClass
  - Gen-Z Network Fabric
    - S3, S4: one Gen-Z port each connected off node (could easily be more)
  - Management
    - Fabric Manager only (might not be typical)
- Fabric Manager
  - Local Fabric
    - B2 Gen-Z bridge: Processor base class (non-boot)
      Direct access, Out-of-Band managed by co-located manager
  - Gen-Z Network Fabric
    - SB2: Gen-Z switch logic integrated in bridge (in-band managed)
    - SoC: Any compatible CPU/SoC
  - Management
    - SoC is co-located Primary Fabric Manager

3.1.1. Grand Plans, Namespaces, and HW Prep

- Primary Fabric Manager runs as application level code on fully privileged hardware (Fabric Manager node)
- Primary Fabric Manager Grand Plan is installed on Fabric Manager Node prior to boot, and calls for Fabric Manager
  - To be run on a dedicated Fabric Manager node
  - To use local access methods to manage its bridge as a co-located Fabric Manager

  - To only acquire control over un-managed components with Manager Type == '1b', per Section *4.1 General Manager Domain Exploration Policies*
  - To maintain a single CID namespace as dictated in the Grand Plan's fabric resource description
  - To acquire management control of all additional Gen-Z components when additional Manager Domains are merged into the namespace
  - To be the only Primary Fabric Manager for the entire fabric namespace
  - To expect a Secondary Fabric Manager may eventually contact it
  - To expect a Composability Manager may eventually contact it
  - To not expect or search for Fabric Directors

- Primary Manager on Large Compute Node runs as system firmware (arbitrary choice) on SoC
- Primary Manager Grand Plan is installed on compute node prior to boot, and calls for the Primary Manager
  - To use local access methods to manage its bridge as co-located Primary Manager
  - To only acquire control over un-managed components with Manager Type == '0b' per Section *4.1 General Manager Domain Exploration Policies*
  - To ignore un-managed components with a FRU-UUID (optional parameter) different from that of bridge B1/SB1.
  - To apply the namespace and topology as defined in the local Grand Plan
  - To expect certain links of certain components are connected to Fabric Switches, and to contact the Fabric Manager of such Fabric Switches only after the Primary Manager's domain has been configured per the local Grand Plan.
  - To negotiate the transfer of management control from the Primary Manager's out-of-band methods to the in-band management of the Primary Fabric Manager which controls the Fabric Switches.

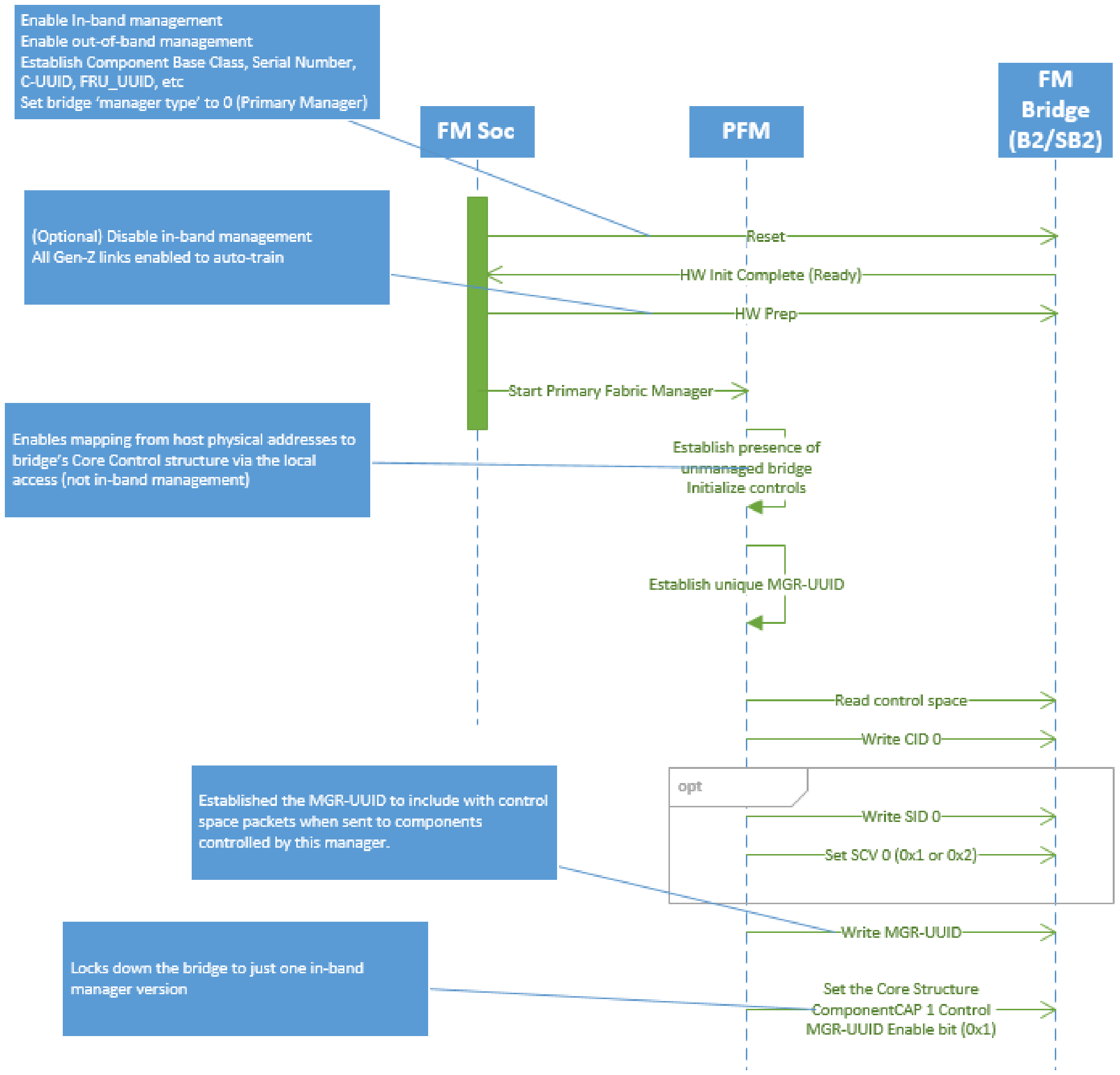
The following components exit the HW Prep phase of initialization with these important settings established:

- Fabric Switches S2 and S5, FAM Drawer Switches S3 and S4, FAM Drawer media controllers C5, C6, C7:
  - Manager Type bit == '1b' (not the default), indicating these components have no Primary Manager expected to configure them before a Primary Fabric Manager discovers them
  - All Gen-Z links enabled to auto-train when peer component at other end of link goes active
- Host bridges B1/SB1 and B2/SB2:
  - Manager Type bit == '0b' (default), indicating these components are not expected to be discovered by a Fabric Manager
  - Local Access (via host interface) enabled for out-of-band management to be performed by host
  - (optional) in-band management disabled, to keep a rogue Fabric Manager from capturing bridge before the host based manager can
  - All Gen-Z links enabled to auto-train when peer component at other end of link goes active

- Compute Node Switches S0, S1, Compute Node media controllers C1, C2, C3, and C4:
  - Manager Type bit == '0b' (default)
  - All Gen-Z links enabled to auto-train when peer component at other end of link goes active

3.1.2. Fabric Manager Bridge (B2/SB2) Configuration

Figure 3-2 Fabric Manager Bridge Configuration Flows illustrates the basic actors and actions needed to configure the Fabric Manager Bridge (B2/SB2) to co-locate the Fabric Manager in the B2/SB2 bridge component.


Enable In-band management
Enable out-of-band management
Establish Component Base Class, Serial Number, C-UUID, FRU_UUID, etc
Set bridge 'manager type' to 0 (Primary Manager)
FM Soc
PFM
FM Bridge (B2/SB2)
(Optional) Disable in-band management
All Gen-Z links enabled to auto-train
Reset
HW Init Complete (Ready)
HW Prep
Start Primary Fabric Manager
Enables mapping from host physical addresses to bridge's Core Control structure via the local access (not in-band management)
Establish presence of unmanaged bridge
Initialize controls
Establish unique MGR-UUID
Read control space
Write CID 0
opt
Write SID 0
Set SCV 0 (0x1 or 0x2)
Established the MGR-UUID to include with control space packets when sent to components controlled by this manager.
Write MGR-UUID
Locks down the bridge to just one in-band manager version
Set the Core Structure ComponentCAP 1 Control MGR-UUID Enable bit (0x1)

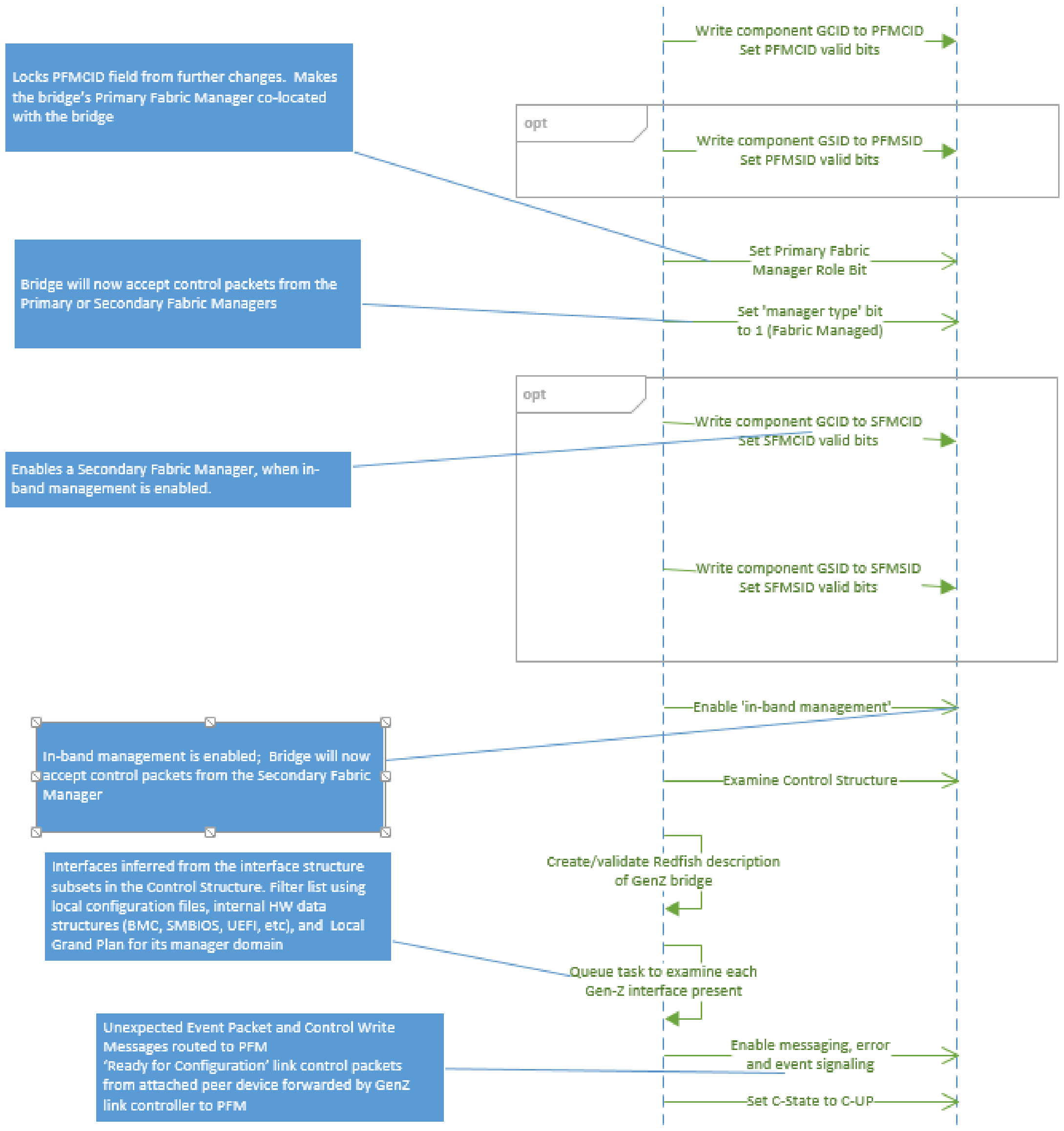

Write component GCID to PFMCID
Set PFMCID valid bits
Locks PFMCID field from further changes. Makes the bridge's Primary Fabric Manager co-located with the bridge
opt
Write component GSID to PFMSID
Set PFMSID valid bits
Set Primary Fabric Manager Role Bit
Bridge will now accept control packets from the Primary or Secondary Fabric Managers
Set 'manager type' bit to 1 (Fabric Managed)
opt
Write component GCID to SFMCID
Set SFMCID valid bits
Enables a Secondary Fabric Manager, when in-band management is enabled.
Write component GSID to SFMSID
Set SFMSID valid bits
Enable 'in-band management'
In-band management is enabled; Bridge will now accept control packets from the Secondary Fabric Manager
Examine Control Structure
Create/validate Redfish description of GenZ bridge
Interfaces inferred from the interface structure subsets in the Control Structure. Filter list using local configuration files, internal HW data structures (BMC, SMBIOS, UEFI, etc), and Local Grand Plan for its manager domain
Queue task to examine each Gen-Z interface present
Unexpected Event Packet and Control Write Messages routed to PFM
'Ready for Configuration' link control packets from attached peer device forwarded by GenZ link controller to PFM
Enable messaging, error and event signaling
Set C-State to C-UP

Figure 3-2 Fabric Manager Bridge Configuration Flows

As fabric components are enumerated and configured, appropriate modifications to the mapping controls within the bridge are made to extend the control plane and data space mappings of the Fabric Manager Node to include the control and data space addresses claimed by those components.

Order of Component Discovery

Once the bridge (B2) is brought to C-UP, the Fabric Manager will proceed to discover, enumerate, and configure Gen-Z components which are found on the Gen-Z links which are exported. The recommended search order is breadth first, which will encounter the following components in the following 'layers', though the order within a given layer is not determined by this policy:

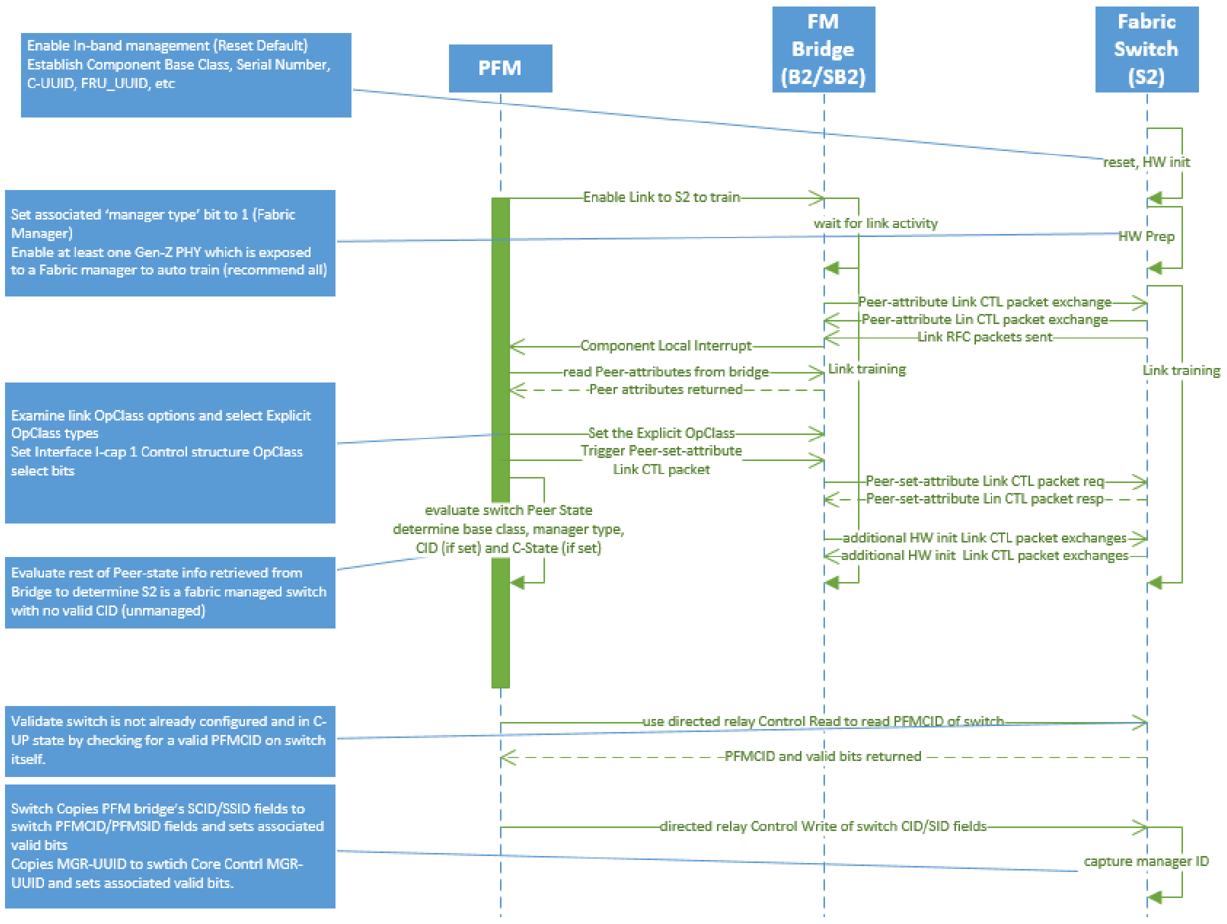
1. Fabric switches S2 and S5

- These components are un-managed, with Manager Type ==1 == 'Fabric Manager', so these components will be queued for configuration by the Fabric Manager

2. FAM Drawer switches S3 and S4, Compute Node Bridge B1/SB1, and Compute node switch S1
    - FAM Drawer components are un-managed, with Manager Type ==1, so these components will be queued for configuration by the Fabric Manager
    - Compute Node components may not be powered, in which case there are no peer-components to investigate…
    - or the Compute Node components may be managed by the Primary Manager on the Compute node, so these components will not be queued for further investigation by the Primary Fabric Manager per Section 4.1, *General Manager Domain Exploration Policies.*
    - The links between S2 and B1/SB1 and between S2 and S1 may be restricted to passing only directed relay Control Write MSGs, per Section *4.4 Securing Boundary Links.*
3. FAM media controllers C5, C6, and C7 via S3, FAM media controllers C5, C6, and C7 via S4
    - The first time these FAM Drawer media controllers are encountered as peer components they are un-managed, with Manager Type ==1, so these components will be configured by the Fabric Manager, per Section 4.1, *General Manager Domain Exploration Policies*.
    - Since these components are encountered during both the S3 and the S4 configuration sequences, all these components will be examined and evaluated twice for configuration status. The first time each is examined, it will be un-managed and the Fabric Manager will queue each for configuration. The second time each is examined, it will be discovered as already managed by the Fabric Manager.

3.1.3. Fabric Switches (S2, S5) Configuration

The first Gen-Z peer components located by the Fabric Manager when it begins Discovery of the fabric outside the local bridge (SB2) are S2 and S5. *Figure 3-3: Fabric Switch S2 Configuration Flows* shows the major actors and actions required for the Fabric Manager to discover and configure Switch S2. Switch S5 configuration is nearly identical actions, most of which can be performed independently from and in parallel with those shown for Switch S2.


PFM
FM Bridge (B2/SB2)
Fabric Switch (S2)
Enable In-band management (Reset Default)
Establish Component Base Class, Serial Number, C-UUID, FRU_UUID, etc
reset, HW init
Enable Link to S2 to train
wait for link activity
HW Prep
Set associated 'manager type' bit to 1 (Fabric Manager)
Enable at least one Gen-Z PHY which is exposed to a Fabric manager to auto train (recommend all)
Peer-attribute Link CTL packet exchange
Peer-attribute Lin CTL packet exchange
Link RFC packets sent
Component Local Interrupt
read Peer-attributes from bridge
Peer attributes returned
Link training
Link training
Examine link OpClass options and select Explicit OpClass types
Set Interface I-cap 1 Control structure OpClass select bits
Set the Explicit OpClass
Trigger Peer-set-attribute Link CTL packet
Peer-set-attribute Link CTL packet req
Peer-set-attribute Lin CTL packet resp
evaluate switch Peer State
determine base class, manager type, CID (if set) and C-State (if set)
additional HW init Link CTL packet exchanges
additional HW init Link CTL packet exchanges
Evaluate rest of Peer-state info retrieved from Bridge to determine S2 is a fabric managed switch with no valid CID (unmanaged)
Validate switch is not already configured and in C-UP state by checking for a valid PFMCID on switch itself.
use directed relay Control Read to read PFMCID of switch
PFMCID and valid bits returned
Switch Copies PFM bridge's SCID/SSID fields to switch PFMCID/PFMSID fields and sets associated valid bits
Copies MGR-UUID to swtich Core Contrl MGR-UUID and sets associated valid bits.
directed relay Control Write of switch CID/SID fields
capture manager ID

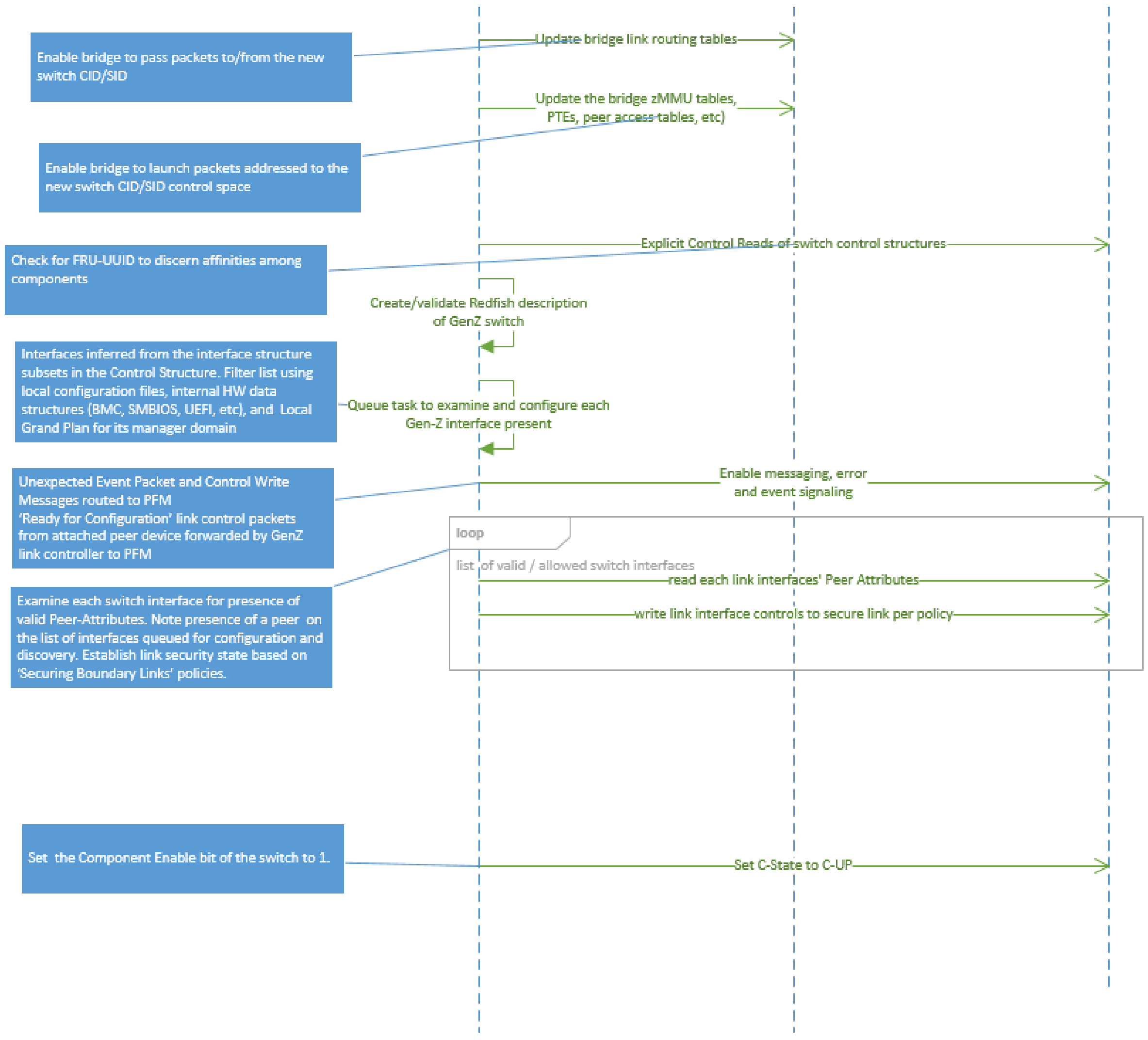

Enable bridge to pass packets to/from the new switch CID/SID
Update bridge link routing tables
Update the bridge zMMU tables, PTEs, peer access tables, etc)
Enable bridge to launch packets addressed to the new switch CID/SID control space
Explicit Control Reads of switch control structures
Check for FRU-UUID to discern affinities among components
Create/validate Redfish description of GenZ switch
Interfaces inferred from the interface structure subsets in the Control Structure. Filter list using local configuration files, internal HW data structures (BMC, SMBIOS, UEFI, etc), and Local Grand Plan for its manager domain
Queue task to examine and configure each Gen-Z interface present
Enable messaging, error and event signaling
Unexpected Event Packet and Control Write Messages routed to PFM 'Ready for Configuration' link control packets from attached peer device forwarded by GenZ link controller to PFM
loop
list of valid / allowed switch interfaces
read each link interfaces' Peer Attributes
write link interface controls to secure link per policy
Examine each switch interface for presence of valid Peer-Attributes. Note presence of a peer on the list of interfaces queued for configuration and discovery. Establish link security state based on 'Securing Boundary Links' policies.
Set the Component Enable bit of the switch to 1.
Set C-State to C-UP

Figure 3-3: Fabric Switch S2 Configuration Flows

The configuration flow for Fabric Switch S5 is almost identical to the configuration of S2, so we won't include a flow diagram for that flow.

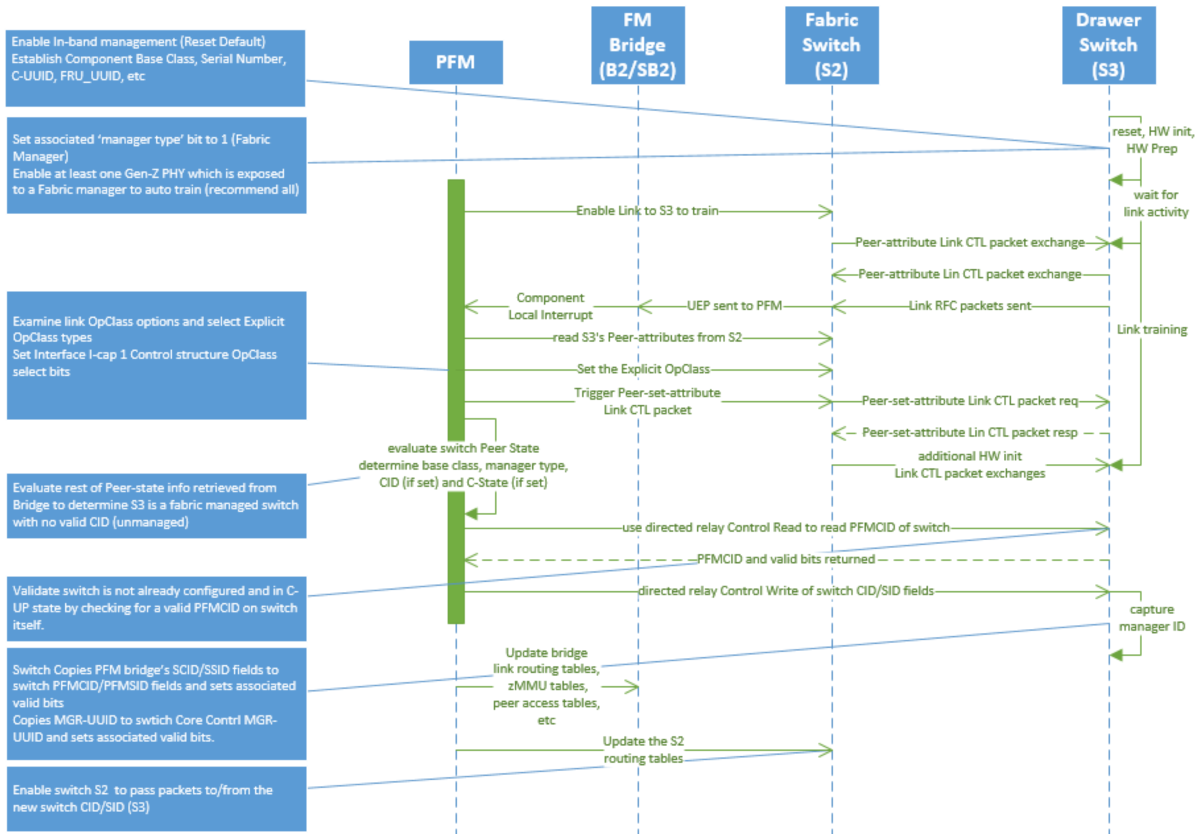
Note that the discussion above under 'Order of Component Discovery' indicates that the Fabric Manager may see the Compute Node's components B1/SB1 and S1 as peer components attached to S2 and S5 respectively. However, these peer components are not candidates for the Fabric Manager to configure, so they will not be queued for such. The links attached to these components will be configured according to Section *4.4 Securing Boundary Links.*

Note that even discovery of the FAM media controllers (C5, C6, and C7) will occur again when they appear as peer-components of the S4 links. Since S4 configuration occurs after S3 configuration, but before configuration of C5, C6, or C7, the Fabric Manager will not recognize that these 3 un-configured media controllers are the same components as already encountered.

3.1.4. FAM Drawer (S3, S4, C7, C5, C6) Configuration

FAM Drawer Enclosure Switch (S3) Configuration

The following figure describes the major actors and actions required to configure the FAM Drawer's Switch S3. Configuration of Switch S4 is very similar, and can be done in parallel.


PFM
FM Bridge (B2/SB2)
Fabric Switch (S2)
Drawer Switch (S3)
Enable In-band management (Reset Default)
Establish Component Base Class, Serial Number, C-UUID, FRU_UUID, etc
Set associated 'manager type' bit to 1 (Fabric Manager)
Enable at least one Gen-Z PHY which is exposed to a Fabric manager to auto train (recommend all)
reset, HW init, HW Prep
wait for link activity
Enable Link to S3 to train
Peer-attribute Link CTL packet exchange
Peer-attribute Lin CTL packet exchange
Component Local Interrupt
UEP sent to PFM
Link RFC packets sent
Link training
Examine link OpClass options and select Explicit OpClass types
Set Interface I-cap 1 Control structure OpClass select bits
read S3's Peer-attributes from S2
Set the Explicit OpClass
Trigger Peer-set-attribute Link CTL packet
Peer-set-attribute Link CTL packet req
Peer-set-attribute Lin CTL packet resp
evaluate switch Peer State
determine base class, manager type, CID (if set) and C-State (if set)
additional HW init
Link CTL packet exchanges
Evaluate rest of Peer-state info retrieved from Bridge to determine S3 is a fabric managed switch with no valid CID (unmanaged)
use directed relay Control Read to read PFMCID of switch
PFMCID and valid bits returned
Validate switch is not already configured and in C-UP state by checking for a valid PFMCID on switch itself.
directed relay Control Write of switch CID/SID fields
capture manager ID
Update bridge link routing tables, zMMU tables, peer access tables, etc
Switch Copies PFM bridge's SCID/SSID fields to switch PFMCID/PFMSID fields and sets associated valid bits
Copies MGR-UUID to swtich Core Contrl MGR-UUID and sets associated valid bits.
Update the S2 routing tables
Enable switch S2 to pass packets to/from the new switch CID/SID (S3)

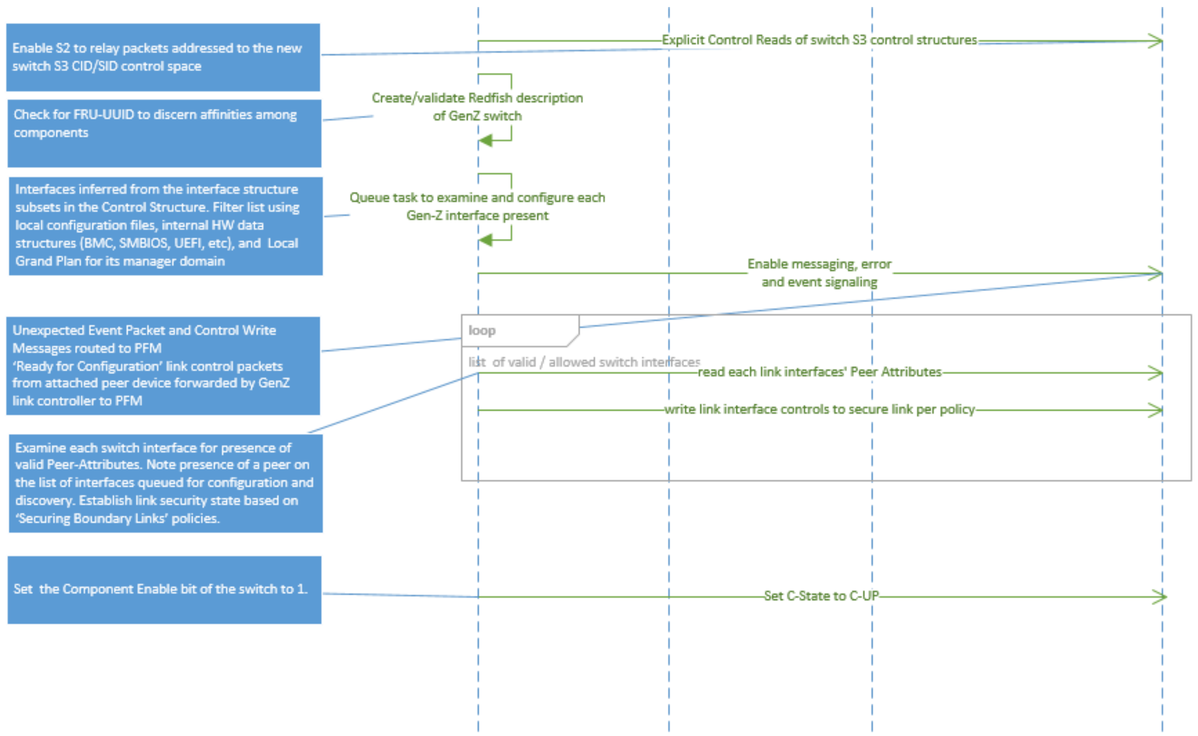
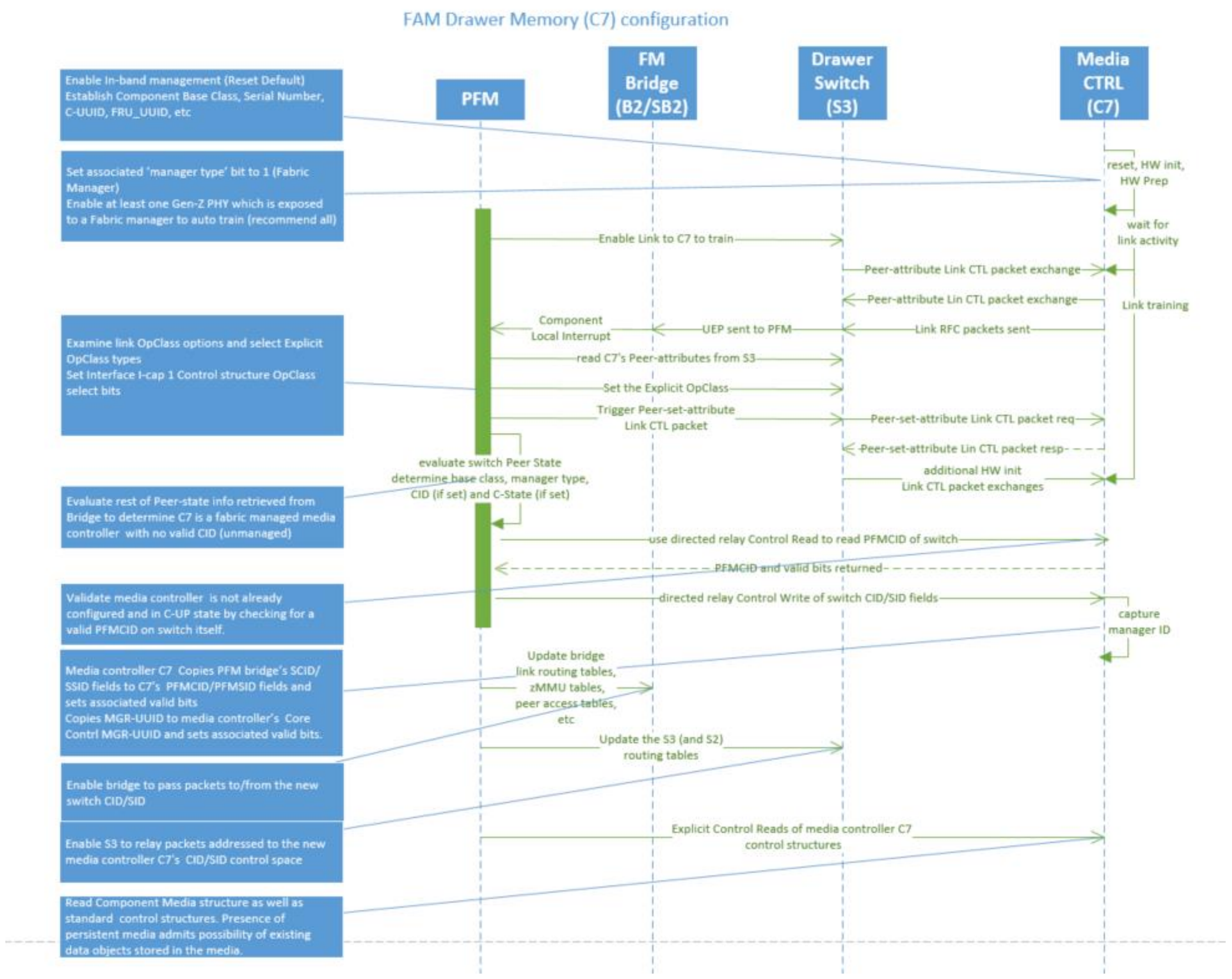

Enable S2 to relay packets addressed to the new switch S3 CID/SID control space
Explicit Control Reads of switch S3 control structures
Check for FRU-UUID to discern affinities among components
Create/validate Redfish description of GenZ switch
Interfaces inferred from the interface structure subsets in the Control Structure. Filter list using local configuration files, internal HW data structures (BMC, SMBIOS, UEFI, etc), and Local Grand Plan for its manager domain
Queue task to examine and configure each Gen-Z interface present
Enable messaging, error and event signaling
Unexpected Event Packet and Control Write Messages routed to PFM 'Ready for Configuration' link control packets from attached peer device forwarded by GenZ link controller to PFM
loop
list of valid / allowed switch interfaces
read each link interfaces' Peer Attributes
write link interface controls to secure link per policy
Examine each switch interface for presence of valid Peer-Attributes. Note presence of a peer on the list of interfaces queued for configuration and discovery. Establish link security state based on 'Securing Boundary Links' policies.
Set the Component Enable bit of the switch to 1.
Set C-State to C-UP

Figure 3-4: FAM Drawer Switch S3 Configuration Flow

The configuration flow for FAM Drawer Switch S4 is almost identical to the configuration of S3, so we will not include that flow. Note that discovery of the FAM media controllers (C5, C6, and C7) will occur again when they appear as peer-components of the S4 links. Since S4 configuration occurs before configuration of C5, C6, or C7, the Fabric Manager will not recognize that these 3 un-configured media controllers are the same components as already encountered.

Memory Configuration

Memory devices hosted behind a ‘media controller’ resident on a switched Gen-Z fabric are a good example of a new class of ‘pooled resources’ to be made available to hosts on a Gen-Z fabric. In this example we look specifically at a persistent memory ‘media controller’ presenting as a base class of “Memory (Explicit OpClass)


FAM Drawer Memory (C7) configuration
PFM
FM Bridge (B2/SB2)
Drawer Switch (S3)
Media CTRL (C7)
Enable In-band management (Reset Default) Establish Component Base Class, Serial Number, C-UUID, FRU_UUID, etc
Set associated ‘manager type’ bit to 1 (Fabric Manager) Enable at least one Gen-Z PHY which is exposed to a Fabric manager to auto train (recommend all)
reset, HW init, HW Prep
wait for link activity
Enable Link to C7 to train
Peer-attribute Link CTL packet exchange
Peer-attribute Lin CTL packet exchange
Link training
Component Local Interrupt
UEP sent to PFM
Link RFC packets sent
Examine link OpClass options and select Explicit OpClass types Set Interface I-cap 1 Control structure OpClass select bits
read C7's Peer-attributes from S3
Set the Explicit OpClass
Trigger Peer-set-attribute Link CTL packet
Peer-set-attribute Link CTL packet req
Peer-set-attribute Lin CTL packet resp
evaluate switch Peer State determine base class, manager type, CID (if set) and C-State (if set)
additional HW init Link CTL packet exchanges
Evaluate rest of Peer-state info retrieved from Bridge to determine C7 is a fabric managed media controller with no valid CID (unmanaged)
use directed relay Control Read to read PFMCID of switch
PFMCID and valid bits returned
Validate media controller is not already configured and in C-UP state by checking for a valid PFMCID on switch itself.
directed relay Control Write of switch CID/SID fields
capture manager ID
Media controller C7 Copies PFM bridge's SCID/SSID fields to C7's PFMCID/PFMSID fields and sets associated valid bits Copies MGR-UUID to media controller's Core Contrl MGR-UUID and sets associated valid bits.
Update bridge link routing tables, zMMU tables, peer access tables, etc
Update the S3 (and S2) routing tables
Enable bridge to pass packets to/from the new switch CID/SID
Enable S3 to relay packets addressed to the new media controller C7's CID/SID control space
Explicit Control Reads of media controller C7 control structures
Read Component Media structure as well as standard control structures. Presence of persistent media admits possibility of existing data objects stored in the media.

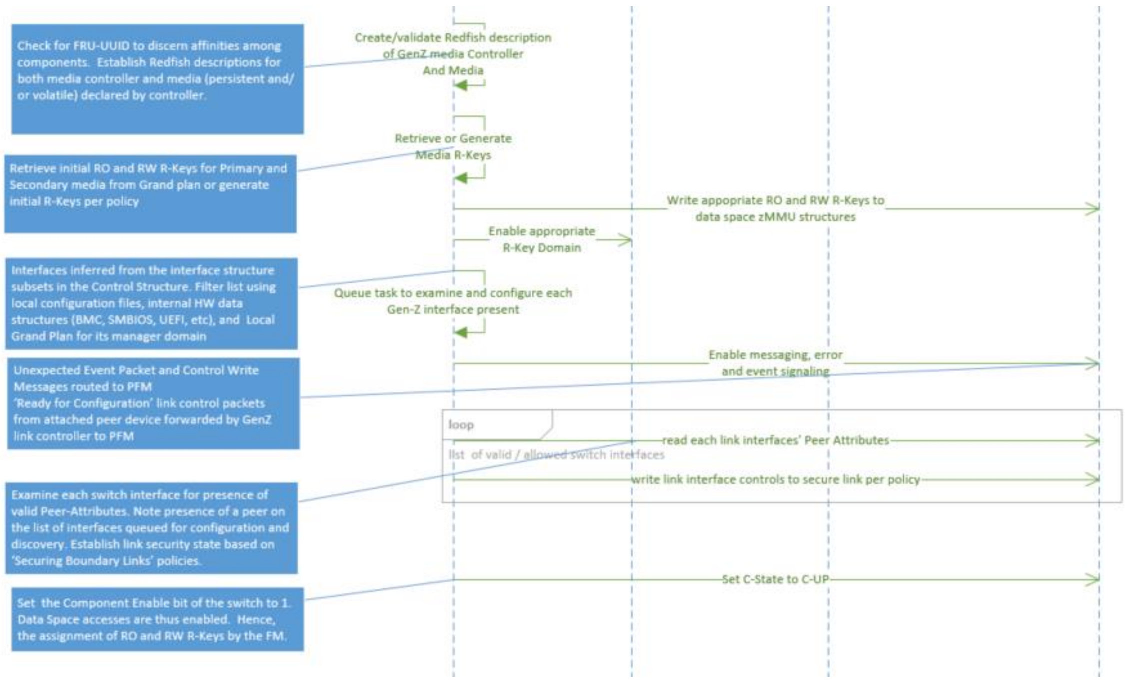

Check for FRU-UUID to discern affinities among components. Establish Redfish descriptions for both media controller and media (persistent and/or volatile) declared by controller.
Create/validate Redfish description of GenZ media Controller And Media
Retrieve initial RO and RW R-Keys for Primary and Secondary media from Grand plan or generate initial R-Keys per policy
Retrieve or Generate Media R-Keys
Write appopriate RO and RW R-Keys to data space zMMU structures
Enable appropriate R-Key Domain
Interfaces inferred from the interface structure subsets in the Control Structure. Filter list using local configuration files, internal HW data structures (BMC, SMBIOS, UEFI, etc), and Local Grand Plan for its manager domain
Queue task to examine and configure each Gen-Z interface present
Unexpected Event Packet and Control Write Messages routed to PFM 'Ready for Configuration' link control packets from attached peer device forwarded by GenZ link controller to PFM
Enable messaging, error and event signaling
loop
list of valid / allowed switch interfaces
read each link interfaces' Peer Attributes
write link interface controls to secure link per policy
Examine each switch interface for presence of valid Peer-Attributes. Note presence of a peer on the list of interfaces queued for configuration and discovery. Establish link security state based on 'Securing Boundary Links' policies.
Set C-State to C-UP
Set the Component Enable bit of the switch to 1. Data Space accesses are thus enabled. Hence, the assignment of RO and RW R-Keys by the FM.

Figure 3-5: FAM Drawer Media Controller C7 Configuration Flow

The configuration flows for the other two media controllers, C5 and C6 are nearly identical to the flow for C7, so we do not show them here. However, as noted earlier the configuration status of each media controller will be examined twice; once when crawling out Switch S3, and once again when crawling out Switch S4. The second examination of media controller S7 has the following flow:

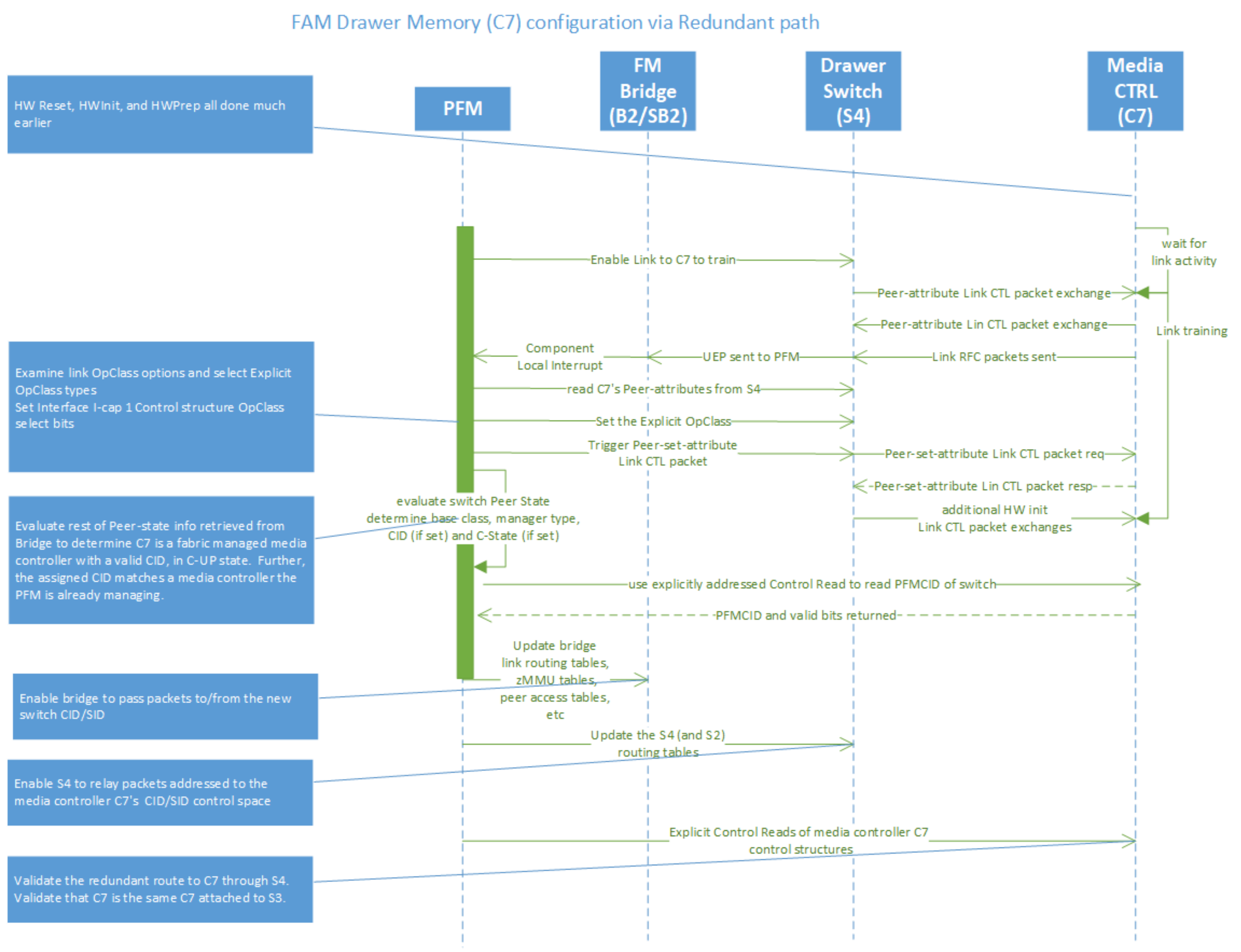

FAM Drawer Memory (C7) configuration via Redundant path
PFM
FM Bridge (B2/SB2)
Drawer Switch (S4)
Media CTRL (C7)
HW Reset, HWInit, and HWPrep all done much earlier
wait for link activity
Enable Link to C7 to train
Peer-attribute Link CTL packet exchange
Peer-attribute Lin CTL packet exchange
Link training
Component Local Interrupt
UEP sent to PFM
Link RFC packets sent
Examine link OpClass options and select Explicit OpClass types
Set Interface I-cap 1 Control structure OpClass select bits
read C7's Peer-attributes from S4
Set the Explicit OpClass
Trigger Peer-set-attribute Link CTL packet
Peer-set-attribute Link CTL packet req
Peer-set-attribute Lin CTL packet resp
additional HW init Link CTL packet exchanges
evaluate switch Peer State determine base class, manager type, CID (if set) and C-State (if set)
Evaluate rest of Peer-state info retrieved from Bridge to determine C7 is a fabric managed media controller with a valid CID, in C-UP state. Further, the assigned CID matches a media controller the PFM is already managing.
use explicitly addressed Control Read to read PFMCID of switch
PFMCID and valid bits returned
Update bridge link routing tables, zMMU tables, peer access tables, etc
Enable bridge to pass packets to/from the new switch CID/SID
Update the S4 (and S2) routing tables
Enable S4 to relay packets addressed to the media controller C7's CID/SID control space
Explicit Control Reads of media controller C7 control structures
Validate the redundant route to C7 through S4.
Validate that C7 is the same C7 attached to S3.

Figure 3-6: FAM Drawer Media Controller C7 (Redundant) Configuration Flow

Note that the Fabric Manager is alerted by the results of the peer attribute query that C7 claims to be in C-UP and have a valid CID. Therefore the Fabric Manager should not attempt to access this component with a direct relay read. Further, the claimed CID matches a media controller this Fabric Manager has previously configured via Switches S2 and S3. Therefore the Fabric Manager enables a (redundant) path via Switches S5 and S4, and proceeds to validate that this component is already configured into its manager domain.

3.1.5. Large Compute Node Configuration

Compute Node Bridge Configuration Flow

The Large Compute Node's Primary Manager Bridge Configuration Flow is very similar to that of the Primary Fabric Manager Configuration Flow as shown in *Figure 3-2 Fabric Manager Bridge Configuration Flows*. The most noticeable difference is that the Primary Manager does not set the Manager Type bit == '1b' == Fabric Manager nor promote itself to be the Primary Fabric Manager before bringing the bridge B1 to C-UP and configuring the rest of the Compute Node's components.

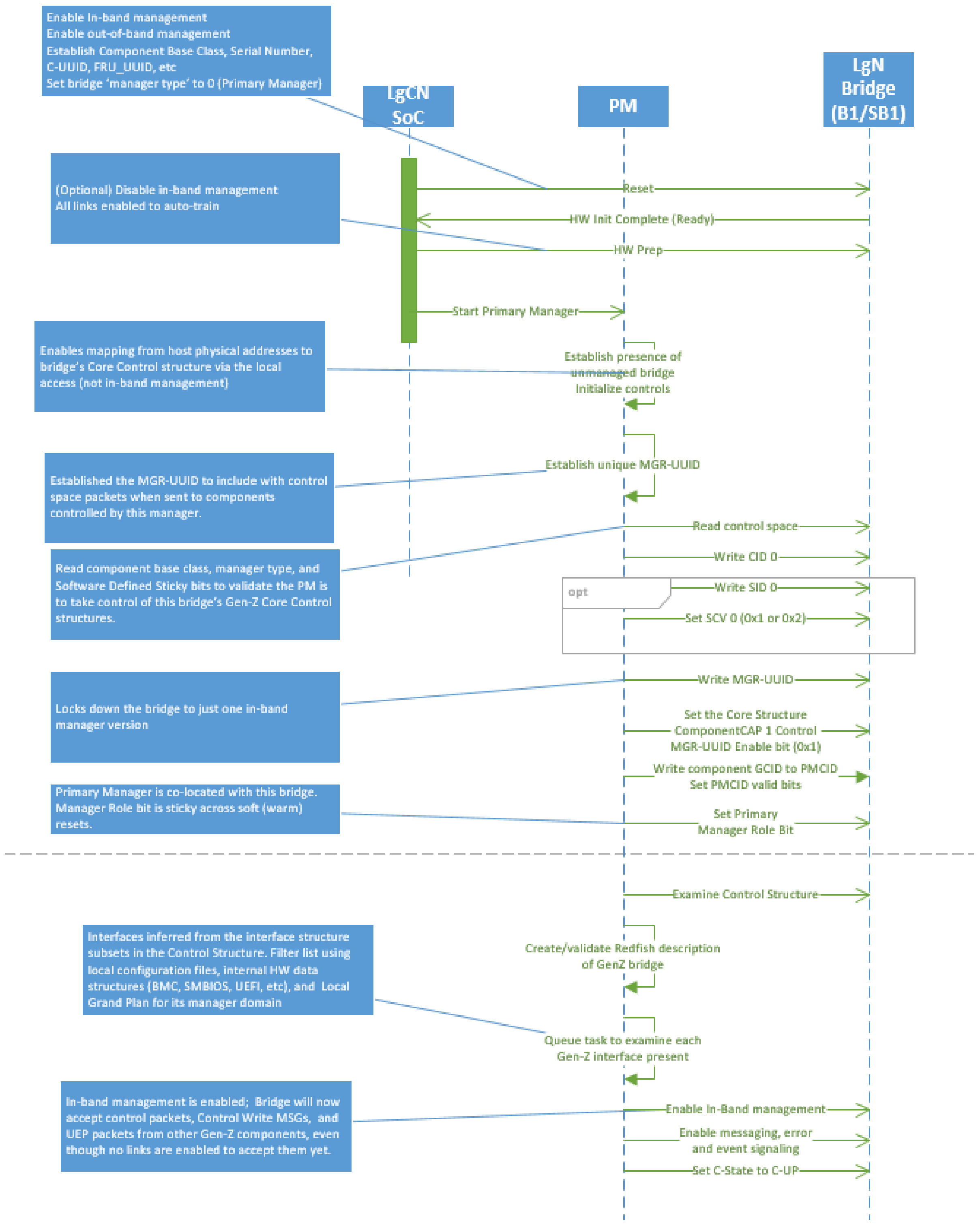

Enable In-band management
Enable out-of-band management
Establish Component Base Class, Serial Number, C-UUID, FRU_UUID, etc
Set bridge 'manager type' to 0 (Primary Manager)
LgCN SoC
PM
LgN Bridge (B1/SB1)
(Optional) Disable in-band management
All links enabled to auto-train
Reset
HW Init Complete (Ready)
HW Prep
Start Primary Manager
Enables mapping from host physical addresses to bridge's Core Control structure via the local access (not in-band management)
Establish presence of unmanaged bridge
Initialize controls
Established the MGR-UUID to include with control space packets when sent to components controlled by this manager.
Establish unique MGR-UUID
Read component base class, manager type, and Software Defined Sticky bits to validate the PM is to take control of this bridge's Gen-Z Core Control structures.
Read control space
Write CID 0
opt
Write SID 0
Set SCV 0 (0x1 or 0x2)
Locks down the bridge to just one in-band manager version
Write MGR-UUID
Set the Core Structure ComponentCAP 1 Control MGR-UUID Enable bit (0x1)
Write component GCID to PMCID
Set PMCID valid bits
Primary Manager is co-located with this bridge. Manager Role bit is sticky across soft (warm) resets.
Set Primary Manager Role Bit
Examine Control Structure
Interfaces inferred from the interface structure subsets in the Control Structure. Filter list using local configuration files, internal HW data structures (BMC, SMBIOS, UEFI, etc), and Local Grand Plan for its manager domain
Create/validate Redfish description of GenZ bridge
Queue task to examine each Gen-Z interface present
In-band management is enabled; Bridge will now accept control packets, Control Write MSGs, and UEP packets from other Gen-Z components, even though no links are enabled to accept them yet.
Enable In-Band management
Enable messaging, error and event signaling
Set C-State to C-UP

Figure 3-7: Compute Node Bridge (B1) Primary Manager Configuration Flow

Once the Primary Manager has brought its bridge to C-UP, it will explore the links off the bridge to discover these components, not necessarily in this order:

- Compute Node Switch S0
- Compute Node Switch S1
- Fabric Switch S2

Configuration of Compute Node Switches S0 and S1

The discovery and configuration flows for Compute Node Switches S0 and S1 are very similar to the flows used by the Primary Fabric Manager to configure Switch S2, which are described in *Figure 3-3: Fabric Switch S2 Configuration Flows.* The major difference is the Switches S0 and S1 will capture the SCID of the bridge B1/SB1 as the Primary Manager CID (PMCID) since the switch's Manager Type bit == '0b', which indicates it is looking for a Primary Manager, rather than a Fabric Manager.

Configuration of Compute Node Media Controllers C1, C2, C3, and C4

During the crawl out of S0 and S1, the Primary Manager will have discovered peer components C1, C2, C3, and C4 attached to links off S0 and S1. C3 will be discovered twice since it has links to both S0 and S1. The configuration flows for C1 through C4 from Switch S0 are essentially as those for the media controller C7 being configured through Fabric Switch S2, as in *Figure 3-5: FAM Drawer Media Controller C7 Configuration Flow*. The attempt to configure C3 a second time through Switch S1 will be very similar to *Figure 3-6: FAM Drawer Media Controller C7 (Redundant) Configuration Flow*. Again, the major difference in these flows for a Primary Manager versus a Fabric Manager is that component will copy the first control write packet's SCID (bridge B1/SB1) to the PMCID field rather than the PFMCID since all these components have the Manager Type bit set to '0b'.

Detection of Manager Domain Collisions at S2 and S5

When the Primary Manager enumerated the links of bridge B1/SB1 and switch S1, the peer attributes of the links attached between B1 to S2 and between S1 to S5 indicated that S2 and S5 were both configured (had CID 0 assigned), were both managed by a Fabric Manager (Manager Type == '1b'), and both are in C-UP state. Since the Primary Manager on the Compute Node did not configure these components, it assumes these components are gateways onto a general Gen-Z fabric. Further, the Primary Manager's local Grand Plan called for the Compute Node to be merged onto this Gen-Z fabric before the OS of the Compute Node is loaded.

Thus, the next task of the Primary Manager is to establish communication with the Primary Fabric Manager in control of the switches S2 and S5. For details of this process, see Section 4.2, *Manager to Manager Communication.* The examples there are based on our Moderate Sized System herein.

Grand Plan and Namespace of the PFM Concerning the Large Compute Node

As stated early in this section, the Fabric Manager running on the Fabric Manager Node is expecting the Large Compute Node to have its own Primary Manager who will do the initial configuration of the node. Per plan, the Fabric Manager will

- **Exchange messages with the Large Compute Node's Primary Manager** as described in the sequences above.
- **Enable control space paths to the Large Compute Node's components,** starting with paths to B1/SB1 and S1, and working through them to C1, C2, C3 and C4 as the Primary Manager enables the Primary Fabric Manager on each.

After the Primary Fabric Manager takes control of the components on the Large Compute Node, it is done with basic Enumeration of Discovery of Gen-Z components of the Moderate Sized System.

- The Primary Fabric Manager should exchange a final message with the Primary Manager indicating it has obtained full access to all the components and assumes responsibility as their Primary Fabric Manager.
- Per policy established by the creator of the Grand Plan, the Primary Fabric Manager may now assign a new value to the Secondary Fabric Manager CID and SID fields, or simply set the values to 'not valid'. Either action will remove manager privileges from the original Primary Manager of the Large Compute Node.
- The Primary Fabric Manager should also remove the original Primary manager's privilege to access the bridge B1's Gen-Z Control Space via the local host interface.

3.1.6. Discovery of Non-Gen-Z Components

While traversing the Core Control CAP 1 structures of the various Gen-Z components found, the Primary and Fabric Managers will have cataloged which components support MCTP over Gen-Z features. MCTP over Gen-Z is the transport and protocol architected to discover and configure non-Gen-Z components found in the Large Compute Node, the Fabric Manager node, the fabric switches' enclosures, and the FAM Drawer complex.

These non-Gen-Z components are not managed by the Primary Manager or the Fabric Manager. However, the Redfish descriptions of the Gen-Z resources that make up the Fabric Manager's Domain will include the relevant MCTP over Gen-Z support capabilities that enable an MCTP compliant manager suite to discover and manage Non-Gen-Z components using MCTP command packets encapsulated over Gen-Z via the Control Write MSG packets. See Section 8.7 MCTP over Gen-Z of the Core Specification.

Enumeration, Discovery and subsequent management of any non-Gen-Z components which are MCTP compliant are beyond the scope of this document. MCTP Management threads that wish to issue MCTP commands (via Gen-Z) to non-Gen-Z components will need to use the Gen-Z Local Node Services' MCTP over Gen-Z entry point, as defined by DMTF.org.

3.2. Gen-Z Management for Data Center Scale

Gen-Z is architected to provide composability at rack, row, and data center scaling. This section provides a reference architecture which can enable Gen-Z Management at data center scale.

The Gen-Z fabric tree shown in *Figure 3-8: Gen-Z Fabric with Management Zones* is representative of how a Gen-Z environment might grow in a datacenter. This diagram mainly depicts the in-band management tree with the keeper of the Grand Plan (the Composability Manager working through the Fabric Directors) as the root. The depth and width of the tree depends on the physical Gen-Z topology.

Gen-Z environment can grow both horizontally as well as vertically in a datacenter. To be able to handle horizontal scalability, the example management design includes a pair of redundant "**Fabric Directors**" (FDs) as shown in the following picture. The **"Fabric Director"** acts as a gateway/broker between the Composability Manager and the different management zones by transporting the messages between the Composability Manager and PFMs. The FDs may be required to translate management commands between whatever transport the upper layer management entities employ and the Gen-Z fabric which connects the various Fabric Managers and their managed components.

To manage a datacenter scale Gen-Z environment efficiently, this example follows a segmented and hierarchical approach for management in a data center. We have already discussed hierarchical management layers to accomplish the grand plan. **Management Zones** are one way segmentation may be achieved. **Multiple Manager Domains logically lumped together create a Management zone.** The Composability Manager deals with Management Zones as a software construct. Management Zones facilitate the co-existence of multiple Fabric Managers. Manager to manager messaging exchanges may include the zone information which is part of the grand plan and (presumably) represented in the Redfish resource models as a logical grouping.

Management Zones are one method to provide scalability, resiliency, and redundancy in a systematic manner. For example, multicast commands can be used to send out universal requests to all zones or specific zones by assigning different Multicast Group IDs (MGIDs) to each zone. Zone managers can offload the CM or FD from the chore of customizing broadly applied configuration changes for each specific component when simple multicast packets won't work.

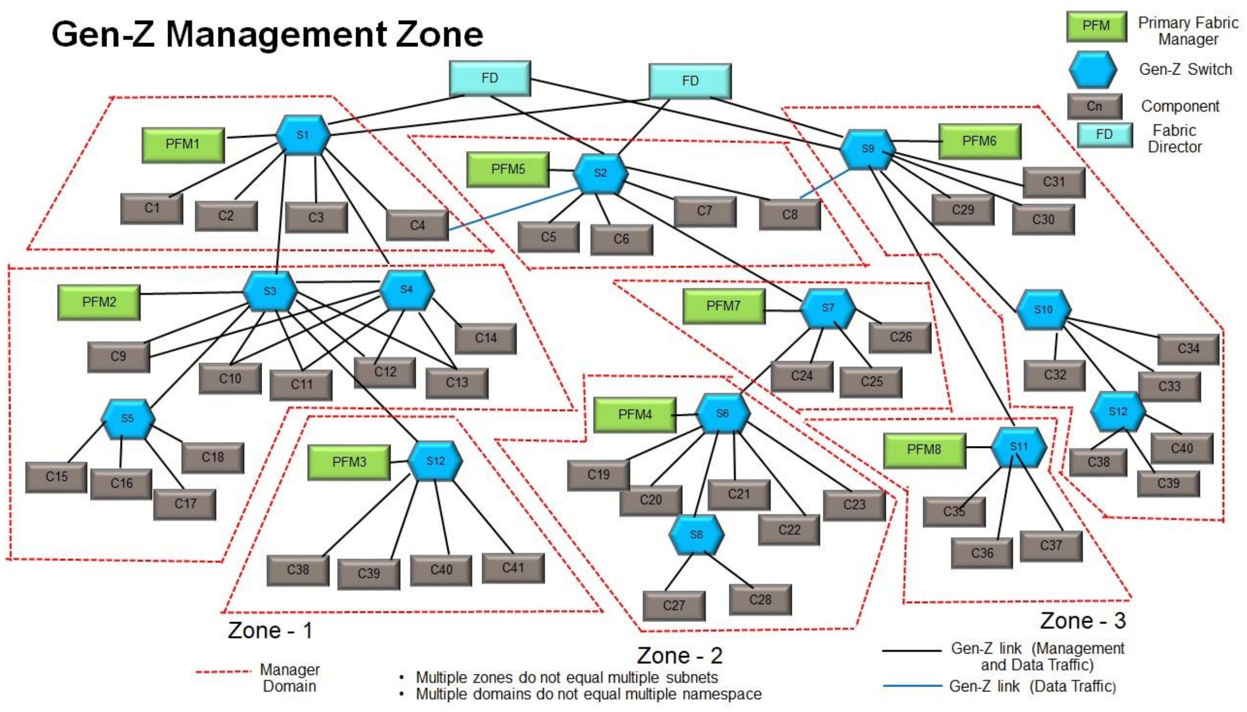

Gen-Z Management Zone
PFM Primary Fabric Manager
Gen-Z Switch
Cn Component
FD Fabric Director
FD
FD
PFM1
PFM2
PFM3
PFM4
PFM5
PFM6
PFM7
PFM8
S1
S2
S3
S4
S5
S6
S7
S8
S9
S10
S11
S12
Zone - 1
Zone - 2
Zone - 3
Manager Domain
• Multiple zones do not equal multiple subnets
• Multiple domains do not equal multiple namespace
Gen-Z link (Management and Data Traffic)
Gen-Z link (Data Traffic)

Figure 3-8: Gen-Z Fabric with Management Zones

3.2.1. Grand Plans and Namespaces

Most management flows outlined in the previous section *3.1 Gen-Z Management for Moderate Sized Systems* remain valid for extreme scale fabrics, but there are some specific scenarios that arise routinely in larger fabrics that were not discussed much, if at all, in this prior use case.

Specifically, the use of multiple Fabric Manager Domains creates at least the following additional namespace and grand plan issues at the data center scale which are relevant to this management specification:

- There is still one global Grand Plan that dictates the desired state of the whole fabric which:
  - If distributed among Multiple Fabric Managers requires global communication and coordination to synchronize the multiple instances of the global plan
  - If kept centrally, also requires similar global communication and coordination to distribute the relevant portions of the global plan to each Fabric Manager
- Most Fabric Managers are not physically adjacent to the Composability Manager and the keeper of the global Grand Plan.
  - Note that the Composability Manager is
  - The Composability Manager (not shown in Figure 3-8) may communicate with the FDs using any appropriate connection; the connections between the FD and higher level manager entities (not shown) such as the Composability Manager are not required to be Gen-Z.
- Multiple Fabric Managers should be enabled to configure their domains in parallel
- Multiple Fabric Manager Domains are likely to boot in an unpredictable order
- All Fabric Manager Domain configurations must be validated to be in compliance with the global Grand Plan before the Domain can be enabled for general participation in the fabric.

As way as example, refer to *Figure 3-8: Gen-Z Fabric with Management Zones* and notice the following Fabric Manager Domains exist in Management Zone-2:

- PFM4: Manager domain consists of
  - Switches S6, S8: fabric switches expecting a fabric manager
  - Components C19-C23, C27, C28: an arbitrary mix of compute, storage, memory or I/O endpoints
- PFM7: Manager domain consists of
  - Switch S7: fabric switch expecting a fabric manager
  - Components C24-C26: an arbitrary mix of compute, storage, memory or I/O endpoints

In general, all the manager domains in Figure 3-8 will boot and go through discovery and enumeration according to their a priori grand plans, much the same as the system described in 3.1 *Gen-Z Management for Moderate Sized Systems*. In this example, PFM7 and its Manager Domain are already up and running and successfully integrated in the global fabric. PFM4 is just booting. In addition, the status of PFM4 at the time it starts discovery and enumeration may be one of the following:

- PFM4 has no a priori local grand plan and hence no namespace plan, no knowledge of the Composability Manager's existence or location, but it does know that it is a Fabric Manager.
- PFM4 has an a priori local grand plan, but it is out of date and thus incorrect.
- PFM4 has a correct a priori local grand plan

In each case above, if PFM4 follows the *Rules of Exploration for Gen-Z Managers*, PFM4 will eventually recognize that S6 is connected to S7, and S7 is a managed switch outside the PFM4 manager domain. PFM4 will then initiate manager to manager messaging, authenticate PFM7, and await knowledge of the global Grand Plan from the Composability Manager.

There are several methods the management software may enable PFM4 to exchange messages with the Composability Manager where ever it resides. The preferred method is for PFM4 to message PFM7 and have PFM7 represent PFM4 as its peer border manager in the PFM7 Redfish tree. If PFM7 adheres to Section 5.1.2 *Required Redfish Support in Gen-Z Managers*, PFM4 can learn of the Composability Manager from the Redfish service of PFM7. The Composability Manager can then access the PFM4 Redfish services to determine the configuration of the PFM4 domain. Until the Composability Manager has sent PFM4 the current Grand Plan details for PFM4's Manager Domain and validated that PFM4 has configured its Domain correctly, PFM4 is not officially integrated into the fabric.

The policies and processes software implements to make this manager domain configuration and validation efficient and robust are beyond the scope of this document.

4. Dealing with Multiple Managers

There are several scenarios wherein two or more in-band manager entities will be present on a Gen-Z fabric and will need to communicate with each other. Some commonly occurring examples are:

- A compute node containing one or more Gen-Z devices is booted, tested, and then joined to an existing Gen-Z switched fabric.
- Two or more stand-alone systems are joined by one or more Gen-Z links. The resulting complex can be of two flavors:
    - One manager is the primary Fabric Manager and presides over a merged / unified Gen-Z CID namespace.
    - Both managers continue as the Primary Manager of their unique name spaces, and they cooperate to expose (probably limited) resources to each other.
        - Exposing even limited resources to another name space requires considerable coordination and cooperation. So much so that we will assume the two managers exchange enough information to create a third, shared name space covering the specific links, ports, and endpoints that are being shared.
        - The shared name space can have no aliases along any of the shared paths.
- Two or more fabric management entities are gaining ownership over and initializing Gen-Z switching components and eventually collide on opposite ends of one or more fabric links.
- One or more compute nodes with Primary Managers are booting even while one or more Fabric Managers are initializing a shared Gen-Z fabric that connects them.

4.1. General Manager Domain Exploration Policies

There are scenarios that call for multiple managers to co-exist and be part of the grand plan. We cover a few of those scenarios in this section of the specification. The architecture specifies rules such that it includes all the basic tenets like plug and play, resiliency, scalability and performance. Management software used in complexes with multiple managers needs to plan on creating MGR-UUIDs per methods defined in the grand plan (Section *2.3.2, Domains, Namespaces, and the Grand Plan*).

4.1.1. Retrieving the Grand Plan

The grand plan includes information on the complete environment, the types of managers expected, and the exploration policies to apply during discovery. Ideally, compute components will be booted with their local version of the grand plan installed such that their Local Node Services or Primary Manager can query this information and act accordingly. Likewise, ideally the keeper of the global Grand Plan and other higher order managers such as the Composability Manager and Resource Managers (when present) are already booted and communicating with each other before the rest of the components in the system are booted, including Fabric Managers (FMs) and Primary Managers (PMs).

However, the discovery rules do cover scenarios where an FM boots without knowledge of the grand plan, or after many other Fabric Managers and Primary Managers have booted.

How the local a priori grand plan is retrieved, and from where are vendor specific, but the format of such data shall be Redfish JSON.

4.1.2. Rules of Exploration for Gen-Z Managers

All in-band Gen-Z manager entities shall follow these rules for determining their role in initialization of Gen-Z devices each encounters while exploring the fabric:

1. Each manager entity shall search for a valid, local a priori grand plan and associated details as soon as practical after launching the manager entity code.
   a. This grand plan shall contain an initial Fabric-UUID value which acts as a CID Namespace identifier.
   b. If no Fabric-UUID is located, or no grand plan can be located, the default Fabric-UUID shall be '0'.
   c. Only non-zero Fabric-UUIDs shall be compared to validate two CID Namespaces are subsets of the same global fabric CID Namespaces.

2. Each manager entity determines if in this instance, in this boot or run state it is destined to act in a 'Fabric Manager' role, or a 'Primary Manager' role. The manager's a priori grand plan normally would establish this role, but there must be failsafe policies embedded in the code.
   a. This moniker defines the manager's expected roles, but it does not dictate how individual managed components identify the manager entity.

3. Each manager entity shall attempt to configure its manager domain in accordance with the namespace, inventory, and topology details contained in said a priori local grand plan.
   a. The information needed

4. Each manager should search for a Composability Manager service as soon as a potential path to such is found.
   a. A depth first search through switches will lead to the quickest discovery of adjacent managers.
   b. Adjacent managers can be contacted as described in *Section 4.2 Manager to Manager Communication* and may exchange Manager Domain status per S*ection 5.3.6 Get Mgr Domain Info Messages.*
   c. A manager currently searching for a connection to a Composability Manager can use the CM Status field of returned Mgr Domain Info Messages to discern those adjacent managers that are potentially a conduit to a suitable Composability Manager.
   d. The Composability Manager is officially a client of the Redfish services provided by the managers sourcing the Redfish resource tree that defines a Manager Domain. The fabric architect is responsible for setting up authorized client lists in the various manager's resource trees in accordance with the Redfish protocol and schema, such that only the desired Composability Manager clients can affect changes on any specific Redfish resource tree.
   e. The fabric architect is responsible for establishing the policies that dictate if, when, and how a Gen-Z Primary or Fabric Manager connects with a given Composability Manager.

When no valid local a priori grand plan exists, or it does not address one or more un-managed components discovered, the following default rules apply. While crawling through the Gen-Z fabric available to it:

5. *only Fabric Managers* shall attempt to own and initialize an un-managed component *whose base class indicates it is a 'fabric switch'.*
6. *only Fabric Managers* shall attempt to own and initialize an un-managed component whose peer state Manager Type is set to '*Fabric Manager'*.
7. *a Fabric Manager* shall not attempt to claim control over any un-managed components on the other side of a fabric managed switch that is not another fabric managed switch unless
   a. that component has its manager type bit set to '*Fabric Manager'*, or
   b. that component is *expected in that Fabric Manager's domain per its grand plan*, or
   c. that component has a non-zero value stored in its *Software Defined Sticky bits which indicates the component is to be Fabric Managed*. See *Section 4.2.3 Component Software Defined Sticky Bits* for the allowed encodings

8. *only Primary Managers* shall attempt to own and initialize an un-managed component whose peer state manager type is set to *'Primary Manager'*
9. *a Primary Manager* shall not attempt to claim control over any un-managed components with Manager Type set to Fabric Manager unless
   a. that component is *expected in that Primary Manager's domain per its grand plan*, or
   b. that component has a non-zero value stored in its *Software Defined Sticky bits which indicates the component is to be Fabric Managed*. See *Section 4.2.3 Component Software Defined Sticky Bits* for the allowed encodings.

Additional rules to be followed:

10. All manager entities which access the Gen-Z fabric through a Gen-Z host bridge component shall set the appropriate Primary Manager Role or Primary Fabric Manager Role bit of the bridge, indicating that the bridge manager is co-located there.
11. When manager domain collisions are detected, managers shall invoke the manager to manager communications protocols defined in *Section 4.2 Manager to Manager Communication* to communicate their knowledge of their local, and the global Grand Plan, to propagate known Gen-Z management services directories, and to negotiate the roles each need to play to complete their respective roles in the grand plan as they currently understand it.

12. If communication between both managers cannot be established or a joint plan cannot be negotiated, the links connecting the two manager domains shall be restricted to relaying only control write messages.
13. Both manager types (Primary and Fabric) may instantiate an 'orphaned component timer' on any unmanaged component that was initially judged to be outside this manager's domain. When such timers expire, the manager shall re-evaluate whether to take control of the unmanaged component at that time.
    a. The implementation of such orphaned component timers and the policies which determine subsequent actions are beyond the scope of this specification.
14. Any of these rules can be overridden by policies included in a local grand plan.
15. No local grand plan shall override the properly authenticated global Grand Plan from the Composability Manager service.
    a. The definition of a 'properly authenticated global Grand Plan' is beyond the scope of this document.

4.2. Manager to Manager Communication

In a Gen-Z Management Zone or across an entire Gen-Z fabric, there can exist multiple Fabric Managers each having ownership of components in a Manager Domain.

Manager to Manager communication is necessary in a fabric for many reasons. The following are typical example use cases:

- Primary Managers and Fabric Managers encounter each other across boundary links.
- Local Node Services (a Gen-Z aware OS or BIOS) queries the Fabric Manager about which Gen-Z resources are available to the server.
- Multiple Fabric Managers encounter each other across boundary links.
- Fabric Managers need to validate their current local grand plan is consistent with the Composability Manager's global Grand Plan.
- Transfer of Component ownership has been negotiated between two managers as part of a Manager Domain collision and resolution process.
- PFM failure detection requires a check of all the PFMs in the zone among themselves and includes any hierarchical managers, such as a Fabric Director.

4.2.1. Initiating the Manager to Manager Communication

See Section 9.3.2 Indirect Fabric Manager Communication of the Gen-Z Core Specification for the requirements of the Control Write MSG functionality in use as a communication protocol between two managers of different manager domains.

The Core Specification dictates all the architected behavior and architected controls of the Control Write MSG packets. Included here are the policies and requirements imposed on the manager software / firmware / hardware which enable multi-manager communications using the Control Write MSG packets.

There are two methods for one manager to target an unreliable Control Write MSG packet at another manager:

a) The sending manager knows the global CID (GCID) of the intended recipient manager and knows that this global address does not conflict with any component's CID within the sending manager's domain.
    a. In this case, the sending manager simply builds an explicitly addressed, unreliable Control Write MSG, using the recipient's known CID as the destination for the packet.
b) The sending manager does not know the GCID of the recipient manager.
    a. In this case, the sending manager must use the directed relay form of the unreliable Control Write MSG, and send it to the boundary component within its own domain that shares the boundary link with the unknown manager domain.

Ingress Port Filter Controls
As explained in Section *2.2.8 Security* and Section *4.4 Securing Boundary Links*, components that have ports which connect to a different manager domain may need to filter the end to end packets which may be relayed across the corresponding link.

- When the peer component is not known to be in the same CID namespace, messages between managers shall be exchanged using directed relay versions of the unreliable Control Write MSG.
- Both ports at each end of the link between the two manager domains thus need to be enabled to accept directed relay packets at the ingress link.
    - To enable a component ingress port to accept a Control OpClass packet with a directed relay variant of the Control OpCode for execution of the directed relay, the ingress port must have the Ingress DR Enable bit set to '1b'.
- Such ports may also be enabled for filtering so they accept *only* directed relay, unreliable Control Write MSGs.
    - To turn off relay of explicitly addressed data space packets, set the Control OpClass Packet Filtering Enable bit to '1b'.
    - To turn off relay of explicitly addressed control space packets, disable packet relay in the ingress port's LPRT and MPRT packet relay tables.
    - To prevent explicitly addressed control packets from targeting a boundary component, install a non-zero MGR-UUID value in the component and enable MGR-UUID checking.
    - To allow *only* directed relay, unreliable Control Write MSGs to ingress at a boundary link, set both the Ingress DR Enable and the Unreliable Control Write MSG Packet Filtering Enable bits to '1b'.

Relaying and Processing Unreliable Control Write MSG packets

Proper execution of the directed relay of the unreliable CWM request requires multiple stages of special handling as it traverses the sender's domain, transitions between domains and then traverses the recipient's domain.

- Phase 1 Address: DR =1, SDR=0, Source and Destination CIDs are valid for the *sender's* domain
    - If the component that creates this Phase 1 Control Write Message is also the target, (i.e. the sender's bridge component is the boundary component), it must launch the packet onto the boundary link using Phase 2 Addressing, per the following Stage 1 processing description.

- The Targeted component of Phase 1 Address packet performs Stage 1 packet processing:
    - Validate packet source is a valid manager
    - Decrypt packet if source Peer Attribute look up returns AEAD_Enable == 1
    - Set SDR of packet SDR = 1, leave DR = 1 (makes packet Phase 2 Address)
    - Convert OpClass from Control OpClass to DR Control OpClass
    - Relay non-encrypted packet to egress port specified in DR_interface field which is a boundary link

- Phase 2 Address: DR =1, SDR = 1, Source and Destination CIDs are invalid for recipient's domain
- Component on receiving end of boundary link performs Stage 2 processing
    - Do no validation of packet source or destination (OpClass == DR Control)
    - Clear DR bit to 0, leave SDR =1 (makes packet Phase 3 address)
    - Set SCID of packet to this component's CID 0
    - Set DCID of packet to this component's current in-band manager
    - Set DR_interface field to ingress port number on which the packet arrived
    - Convert OpClass from DR Control OpClass back to Control OpClass
    - Encrypt packet if destination Peer Attribute look up returns AEAD_Enable == 1

- Relay to egress port specified for current in-band manager's CID
    - If manager is co-located with receiving component, treat packet as if it arrived in this form on original ingress port

A final requirement about manager to manager communications using Control Write MSG packets:

- When the Control Write MSG packet may transit out of the CID namespace of the initial sender, all CWM packets used for manager to manager communications shall use the unreliable Control Write MSG OpCode, and there will be no standalone acknowledgement returned.

4.2.2. Control Write MSG Example

In the following example, refer to *Figure 3-1 Moderate Sized Gen-Z System* for the definition of the various components used. As shown below in *Figure 4-1: Control Write MSG Processing Across Manager Domains*, the following is a summary of the important packet manipulations required to route a directed relay, unreliable Control Write MSG packet from the Primary Manager on Bridge B1 to the Primary Fabric Manager on Bridge B2.

- The Primary Manager creates an unreliable Control Write MSG and uses directed relay because it does not know the CID of the Primary Fabric Manager. It does know that the peer component on the other end of Switch S1's Port 21 is a member of the Primary Fabric Manager's Domain.
- The CWM packet is in Phase 1 addressing state (DR ==1, SDR ==0; CIDs and DR-interface fields are correct for Primary Manager Domain) as it enters into processing stage 1 at Component S1.
- The Responder sends the packet out the designated port (in this case Port 21) sets SDR == 1 and OpClass to DR Control so the peer component on the other end knows it is a Phase 2 address.
- In Stage 2 processing, Component S5 knows to ignore SCID, DCID, and DR-Interface and therefore modifies the packet fields such that it is now Phase 3 addressing (DR == 0, SDR == 1, OpClass = Control, and CIDs and DR-interface fields are correct for the Primary Fabric Manager Domain) and relays the packet to its manager (Bridge B2) in the recipient's domain.
- Finally, in Stage 3 the manager in the new domain reads the packet. At this point, only the original payload of the packet remains. Everything else has been modified. The Phase 3 addressing informs the recipient manager of all details it needs to craft a reply message in the form of a directed relay, unreliable Control Write MSG sent back to Component S5 and then to be relayed out Port 4.

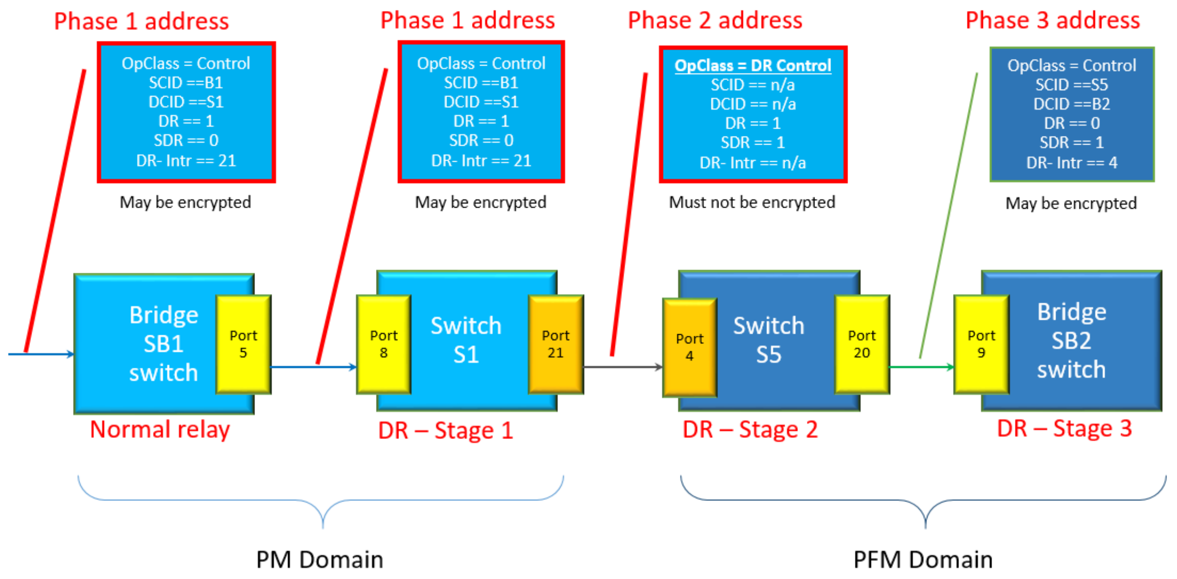


Figure 4-1: Control Write MSG Processing Across Manager Domains

Table 4-1: Directed Relay Control Write MSG Decoding

	Input Port Filter Bits		**Packet State Decode at Ingress Port**				**Action taken by component**
Agent/ Port	Ingress DR Enable	Unreliable Control Write MSG Filter Enable	OpClass	DR	SDRs	DCID == CID 0 of component?	
B1 / -	n/a	n/a	Control	n/a	n/a	0	Normal relay: packet relayed to Port 5 based on SSDT or MSDT lookup at Requestor B1
S1 / 8	1	0	Control	1	0	1	DR = Stage 1: relay to port 21; set SDR = 1
S5 / 4	1	1	DR Control	1	1	n/a	DR – Stage 2: modify packet, send to manager
B2 / 9	0	0	Control	0	1	1	DR – Stage 3: process packet as recipient

4.2.3. Component Software Defined Sticky Bits

The Gen-Z Core Specification Section 8.15.4 defines software defined sticky bits (Software-Defined Management Bits) that retain their values across warm resets. They are intended to be used by management software to leave ‘management state’ hints in the hardware for the next manager to read. The following describes the requirements and restrictions this Gen-Z Management Architecture places on the use of these bits:

- Per each Gen-Z interface, there are 2 Interface Software defined sticky bits that may be used to indicate to managers the prior state of the interface just before it was soft reset.
    a. This specification currently has no restrictions on how Gen-Z managers may use these two sticky bits
- There are 8 software defined sticky bits in the Core Control CAP 1 Control Structure which may be used to indicate to managers the state of the component just before it was soft reset. They are denoted Software-Defined Management Bit x (SDM Bit$_x$), where x = 0 to 7.
    a. Gen-Z managers (in-band and out-of-band) compliant with this management specification shall recognize the following encodings of these 8 Software Defined Sticky bits:

        The 8 bits are divided into two sub-fields within the byte

        - The four bits SDM Bit$_{0-3}$ are reserved for use of manager to manager communications per the following encodings
            - 0x0: no information retained across warm reset (default)
            - 0x1: upon reset, component is to be managed by Fabric Manager
            - 0x2: upon reset, component is to be managed by Primary Manager
            - 0x3-0xf: reserved for future manager to manager handoff protocols
        - The four bits SDM Bit$_{4-7}$ are reserved for general software management use. The values of these bits shall have no impact on the encoded meaning of the first 4 bits.

- Refer to Section *4.1 General Manager Domain Exploration Policies* (above) for the requirements dictating when to set and when to check the Software Defined Stick Bits.

4.3. Transfer of Ownership Between Gen-Z Managers

4.3.1. Explicit Ownership Transfer

An existing manager (in-band or out-of-band) can write the CID of the new in-band manager into one of the in-band manager CID fields [PMCID | PFMCID/PFMSID | SFMCID/SFMSID], set the associated valid bits and adjust the ‘manager type’ bit accordingly. This document is focused on in-band management policies and requirements, so the following examples of scenarios in which ownership is transitioned only include hand-offs between two in-band managers, both of whom operate within the same unified CID namespace.

- A Primary Manager is handing over control of a component in the C-UP state to a known Fabric Manager or a different, known Primary Manager
- A Fabric Manager is handing over control of a component in C-UP state to a different known Fabric Manager
- A Primary Manager is handing over control of a component in C-CFG state to a different known Fabric Manager
- Primary Manager which is co-located with a component in the C-CFG state is handing over control to a known Fabric Manager

In the list above, the term 'known Manager' indicates that the new Manager's Component ID (CID) and Subnet ID (SID) are known to the original manager who is handing over control of the component. Further, that expression also indicates that the current manager and the future manager both belong to the same global CID Namespace. However, both managers are not required to have the same MGR-UUID.

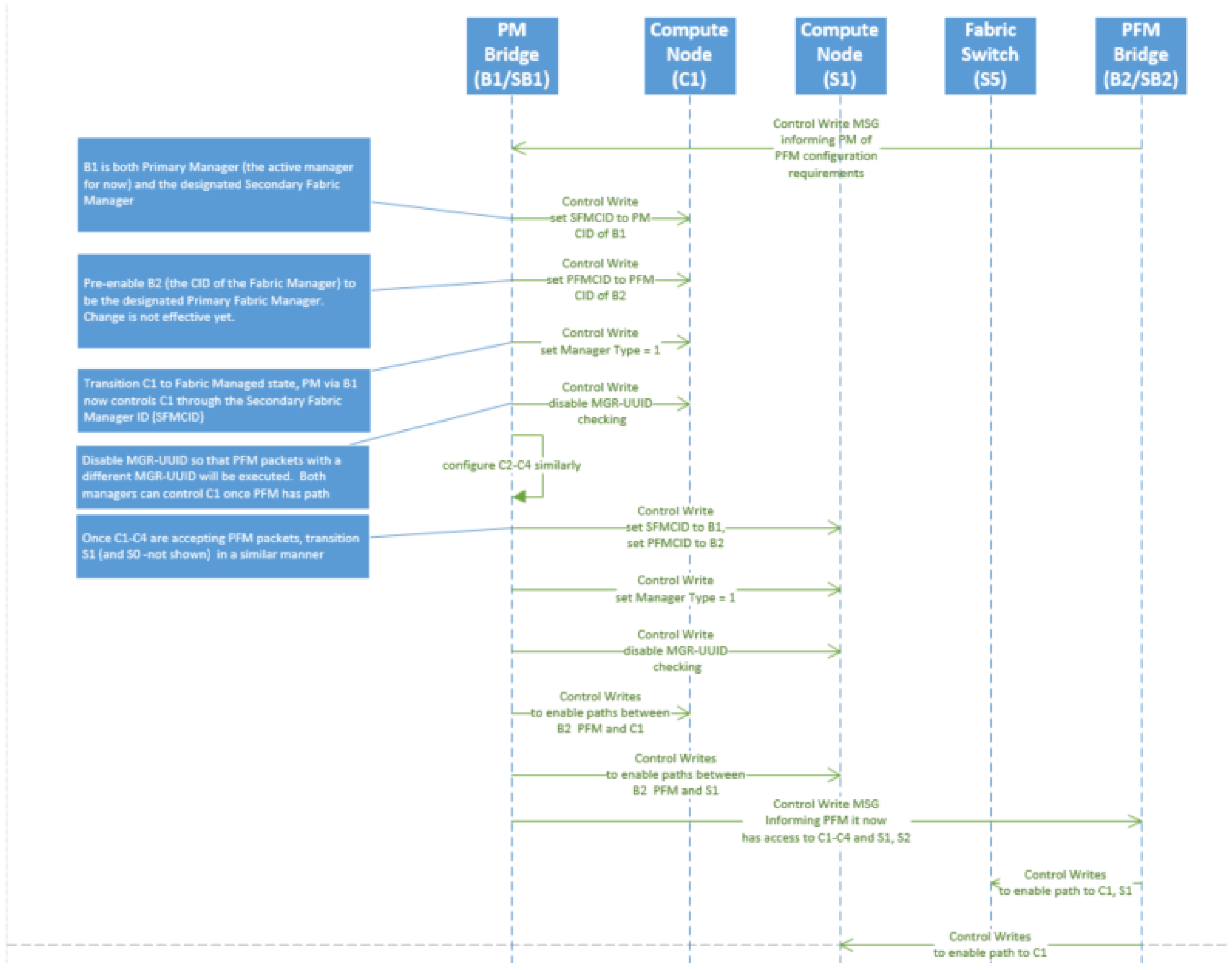
- If the original manager and the new manager do not use the same MGR-UUID, the original manager shall disable MGR-UUID validation so that both managers may share control during the transition period. (See Sections 8.1.1 and 8.14.4 and the Core Cap 1 Control structure in Table 8-8 of the Core Specification for details about MGR-UUIDs.)
- If the original manager needs to disable MGR-UUID validation, it is highly recommended it only do so after completely disabling any ingress ports on the component which attach to or may attach to other Gen-Z components not a member of the original manager's domain, or the new manager's domain.

4.3.1.1. Primary Manager Hand-off to a Known Primary Fabric Manager

In the following example, refer to *Figure 3-1 Moderate Sized Gen-Z System* for the definition of the various components used. The diagram will show only the actions involving S1 and C1, but the configuration changes for C2, C3, C4, and S0 will be essentially the same.

Initial conditions for C1 and S1 (also valid for C2, C3, C4, and S0)

- The Primary Manager PMCID field is set and valid, as is the MGR-UUID.
- The Manager Type bit is equal to '0', indicating the PMCID field must match the DCID of any Control OpClass packets before such packets can be executed.
- The Primary Manager has learned the CID (and SID if appropriate) of the new manager to which it is handing off the component.
- The Primary Manager has determined the new manager's MGR-UUID value will be different from its own.


PM Bridge (B1/SB1)
Compute Node (C1)
Compute Node (S1)
Fabric Switch (S5)
PFM Bridge (B2/SB2)
Control Write MSG informing PM of PFM configuration requirements
Control Write set SFMCID to PM CID of B1
Control Write set PFMCID to PFM CID of B2
Control Write set Manager Type = 1
Control Write disable MGR-UUID checking
configure C2-C4 similarly
Control Write set SFMCID to B1, set PFMCID to B2
Control Write set Manager Type = 1
Control Write disable MGR-UUID checking
Control Writes to enable paths between B2 PFM and C1
Control Writes to enable paths between B2 PFM and S1
Control Write MSG Informing PFM it now has access to C1-C4 and S1, S2
Control Writes to enable path to C1, S1
Control Writes to enable path to C1

B1 is both Primary Manager (the active manager for now) and the designated Secondary Fabric Manager

Pre-enable B2 (the CID of the Fabric Manager) to be the designated Primary Fabric Manager. Change is not effective yet.

Transition C1 to Fabric Managed state, PM via B1 now controls C1 through the Secondary Fabric Manager ID (SFMCID)

Disable MGR-UUID so that PFM packets with a different MGR-UUID will be executed. Both managers can control C1 once PFM has path

Once C1-C4 are accepting PFM packets, transition S1 (and S0 -not shown) in a similar manner

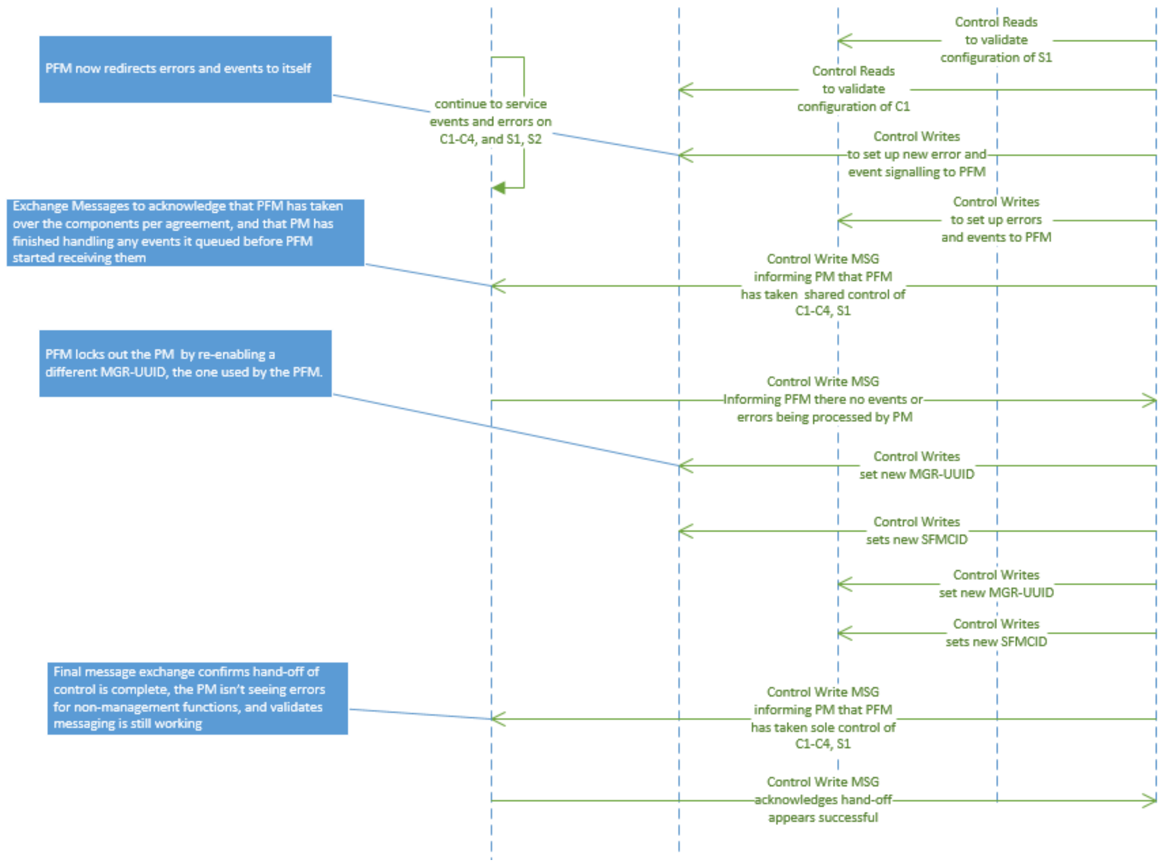

PFM now redirects errors and events to itself
continue to service events and errors on C1-C4, and S1, S2
Exchange Messages to acknowledge that PFM has taken over the components per agreement, and that PM has finished handling any events it queued before PFM started receiving them
PFM locks out the PM by re-enabling a different MGR-UUID, the one used by the PFM.
Final message exchange confirms hand-off of control is complete, the PM isn't seeing errors for non-management functions, and validates messaging is still working
Control Reads to validate configuration of S1
Control Reads to validate configuration of C1
Control Writes to set up new error and event signalling to PFM
Control Writes to set up errors and events to PFM
Control Write MSG informing PM that PFM has taken shared control of C1-C4, S1
Control Write MSG Informing PFM there no events or errors being processed by PM
Control Writes set new MGR-UUID
Control Writes sets new SFMCID
Control Writes set new MGR-UUID
Control Writes sets new SFMCID
Control Write MSG informing PM that PFM has taken sole control of C1-C4, S1
Control Write MSG acknowledges hand-off appears successful

Figure 4-2: Explicit Ownership Transfer Example Flow

The content and format of the synchronizing messages sent between managers to initiate and confirm management control handoff is expected to be part of the Redfish protocol, and not addressed in this document.

4.3.1.2. Primary Manager Handoff to another known Primary Manager

The steps required of an existing Primary Manager to hand-off to a different, known Primary Manager are nearly the same as above. The original Primary Manager will install itself as the Secondary Fabric Manager, enable the desired new Primary Manager as the Primary Fabric Manager, and then set the Manager Type bit = '1b' to enable both managers as 'Fabric Managers'.

- Since there is no secondary Primary Manager, the original Primary Manager should not simply replace the PMCID field value with the CID of the new Primary Manager, as this method of transfer of control is not reversible if the new Primary Manager fails to complete the take-over.
- Once the new manager has control as the Primary Fabric Manager, it can choose to install itself as the new Primary Manager and clear the Manager Type bit to '0b'. Doing this will convert the component back to 'Primary Managed', and lock out the original Primary Manager at the same time.

4.3.1.3. Fabric Manager Handoff to another Fabric Manager

The steps required of an existing Primary Fabric Manager to hand-off to a different, known Primary Fabric Manager are also similar to *Figure 4-2: Explicit Ownership Transfer Example Flow,* with appropriate notation changes since the original manager is not the Primary Manager, but rather the Primary Fabric Manager.

The original Primary Fabric Manager will place itself into the Secondary Fabric Manager's slot (SFMCID) and write the new Primary Fabric Manager's ID into the PFMCID | PFMSID fields.

4.3.1.4. Co-Located Primary Manager Hand-off to Fabric Manager

There are small differences in the steps required for a Co-Located Primary Manager to hand over ownership of its bridge component to a known Fabric Manager. The process is generally the same, except that the original Primary Manager needs to alter the Primary Manager Role and Primary Fabric Manager Role control bits.

Again, in the following example, refer to *Figure 3-1 Moderate Sized Gen-Z System* for the definition of the various components used.

Initial conditions for Bridge B1/SB1:

- The Primary Manager PMCID field is set and valid, as is the MGR-UUID.
- The Primary Manager Co-located bit is set, and direct writes to the PMCID field are thus prohibited.
- The Manager Type bit is cleared to '0', indicating the PMCID field must match the DCID of any Control OpClass packets before such packets can be executed.
- The Primary Manager has learned the CID (and SID if appropriate) of the new manager to which it is handing off the component.
- The Primary Manager has determined the new manager's MGR-UUID value will be different from its own.
- The Primary Manager is still accessing the bridge B1's Control Space using Local Access (out-of-band) permissions.
- In-band Management is currently enabled so bridge can manage other components

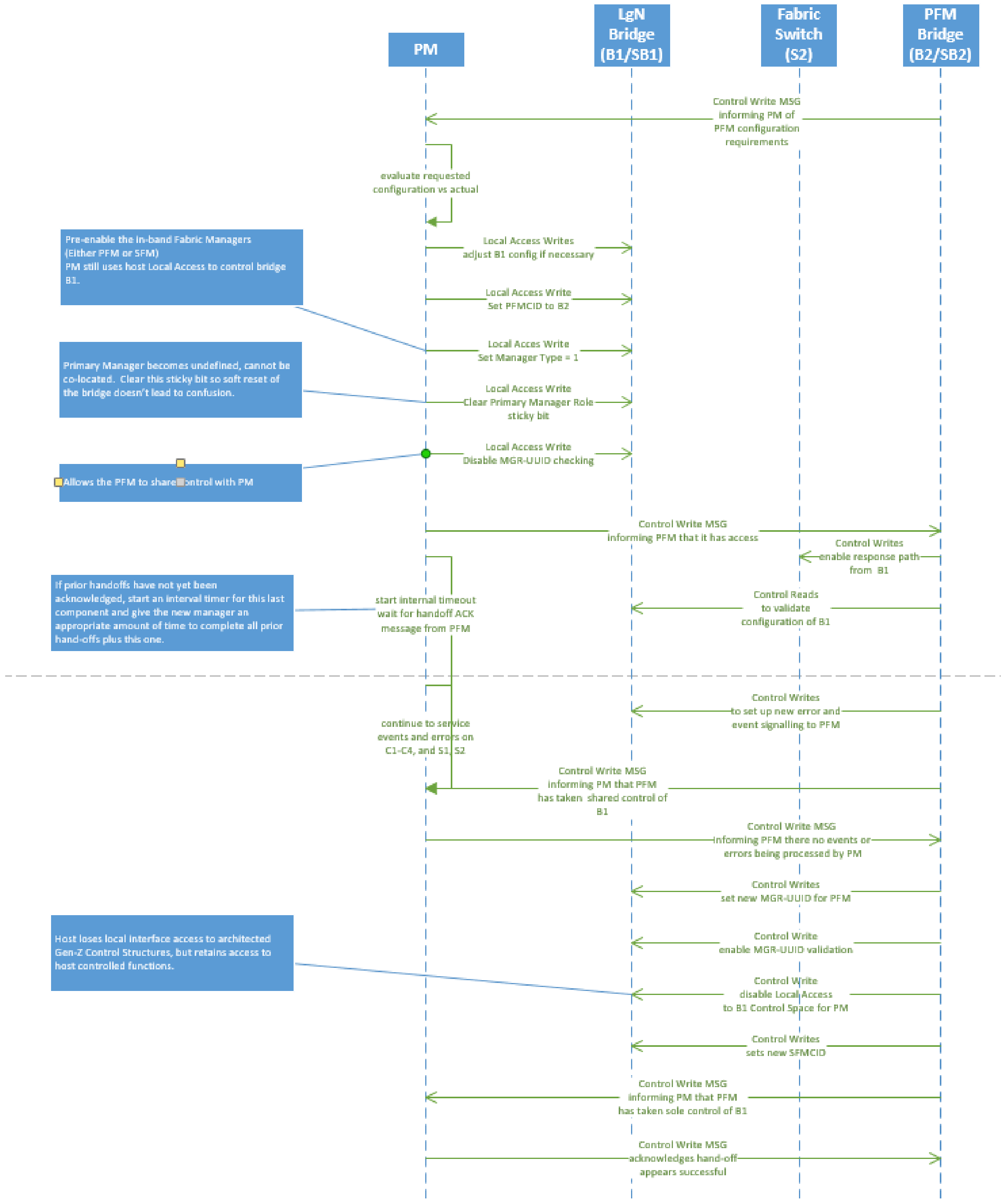

PM
LgN Bridge (B1/SB1)
Fabric Switch (S2)
PFM Bridge (B2/SB2)
Control Write MSG informing PM of PFM configuration requirements
evaluate requested configuration vs actual
Pre-enable the in-band Fabric Managers (Either PFM or SFM) PM still uses host Local Access to control bridge B1.
Local Access Writes adjust B1 config if necessary
Local Access Write Set PFMCID to B2
Local Acces Write Set Manager Type = 1
Primary Manager becomes undefined, cannot be co-located. Clear this sticky bit so soft reset of the bridge doesn't lead to confusion.
Local Access Write Clear Primary Manager Role sticky bit
Local Access Write Disable MGR-UUID checking
Allows the PFM to share control with PM
Control Write MSG informing PFM that it has access
Control Writes enable response path from B1
If prior handoffs have not yet been acknowledged, start an interval timer for this last component and give the new manager an appropriate amount of time to complete all prior hand-offs plus this one.
start internal timeout wait for handoff ACK message from PFM
Control Reads to validate configuration of B1
Control Writes to set up new error and event signalling to PFM
continue to service events and errors on C1-C4, and S1, S2
Control Write MSG informing PM that PFM has taken shared control of B1
Control Write MSG informing PFM there no events or errors being processed by PM
Control Writes set new MGR-UUID for PFM
Control Write enable MGR-UUID validation
Host loses local interface access to architected Gen-Z Control Structures, but retains access to host controlled functions.
Control Write disable Local Access to B1 Control Space for PM
Control Writes sets new SFMCID
Control Write MSG informing PM that PFM has taken sole control of B1
Control Write MSG acknowledges hand-off appears successful

Figure 4-3: Explicit Ownership Handoff of Primary Manager's Bridge

4.3.2. Abdication of Ownership

Manager Transfer of Ownership by Abdication makes sense when two different manager domains in two different CID namespaces are merged. For example:

- A large fabric (Fabric A) is up and running, and another running fabric (Fabric B) which is lower in priority with a smaller namespace is to be merged into it.
- The lower priority manager's (Manager B) own local Grand Plan contains no details about the Fabric Manager Domain's namespace nor a valid CID at which to message the FM.
- The higher priority domain stays up and running, and the lower priority domain is taken offline by its manager and the components are all reset. The lower priority manager (Manager B) does not attempt to re-discover and re-configure its domain after the global reset. Domain Manager A then sees the reset components from Fabric B come online as unmanaged components ready to be added to its domain.

All transfer of ownership scenarios that depend upon the new manager capturing the appropriate manager ID fields (PMCID or PFMCID) of un-managed components require the following:

- At least one link of the component shall be connected to a peer that is, or will be, managed by the new manager, AND shall be enabled to auto-train after the component is reset
- The abdicating manager shall set the appropriate code into the Software Defined Sticky Bits so that the unknown manager knows that control of the component has been abdicated by the Primary Manager. See *4.2.3 Component Software Defined Sticky Bits* for the architected encodings.
- The abdicating manager shall issue a Warm Component Reset to the device.

4.3.2.1. *Primary Manager Abdicates Control to Fabric Manager*

The following sequence outlines a method for using Software Defined Sticky bits to flag the reset component as an exception to the standard discovery process it would normally see.

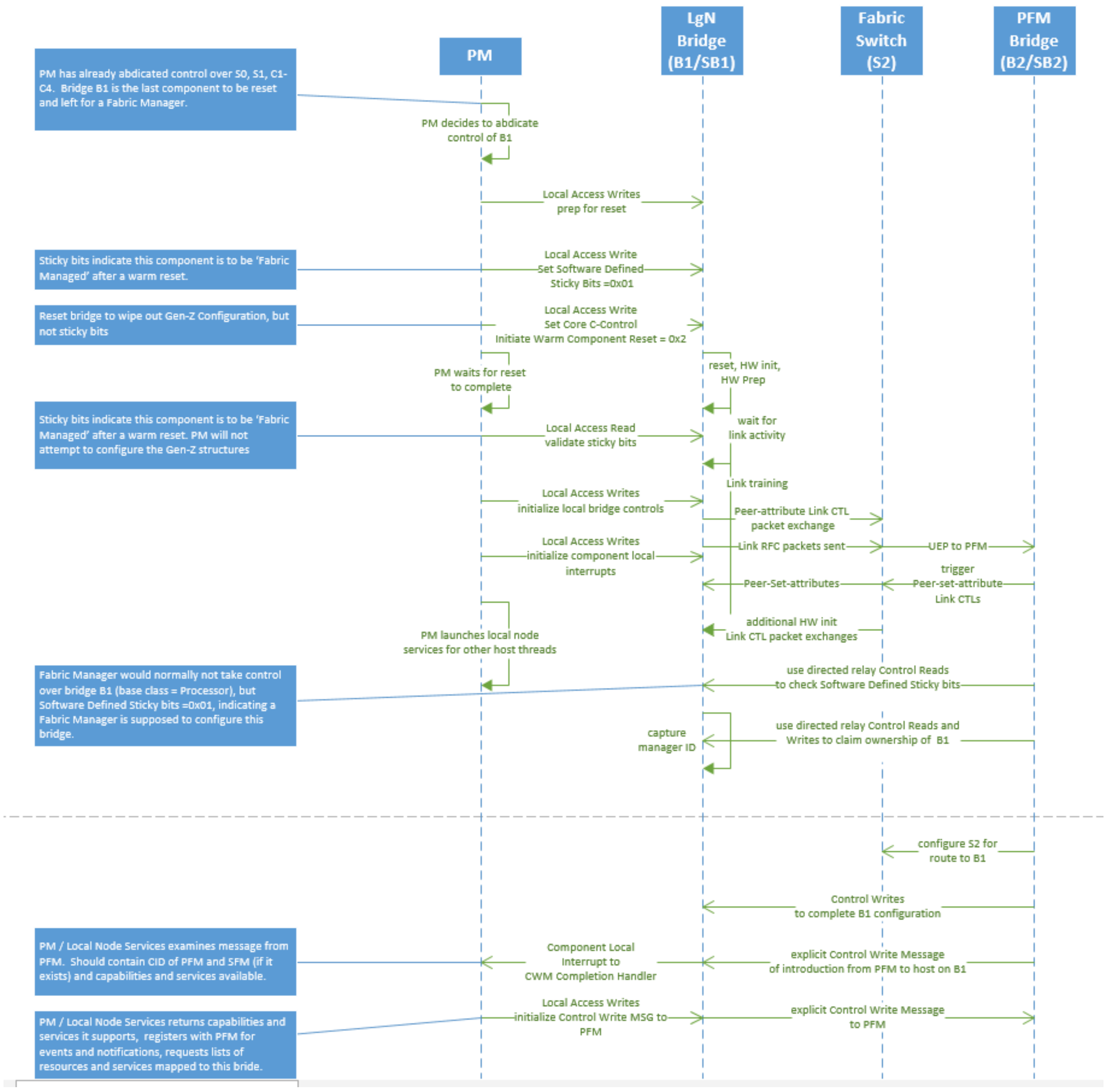

PM
LgN Bridge (B1/SB1)
Fabric Switch (S2)
PFM Bridge (B2/SB2)
PM has already abdicated control over S0, S1, C1-C4. Bridge B1 is the last component to be reset and left for a Fabric Manager.
PM decides to abdicate control of B1
Local Access Writes prep for reset
Sticky bits indicate this component is to be 'Fabric Managed' after a warm reset.
Local Access Write Set Software Defined Sticky Bits =0x01
Reset bridge to wipe out Gen-Z Configuration, but not sticky bits
Local Access Write Set Core C-Control Initiate Warm Component Reset = 0x2
PM waits for reset to complete
reset, HW init, HW Prep
Sticky bits indicate this component is to be 'Fabric Managed' after a warm reset. PM will not attempt to configure the Gen-Z structures
Local Access Read validate sticky bits
wait for link activity
Link training
Local Access Writes initialize local bridge controls
Peer-attribute Link CTL packet exchange
Local Access Writes initialize component local interrupts
Link RFC packets sent
UEP to PFM
trigger Peer-set-attribute Link CTLs
Peer-Set-attributes
PM launches local node services for other host threads
additional HW init Link CTL packet exchanges
Fabric Manager would normally not take control over bridge B1 (base class = Processor), but Software Defined Sticky bits =0x01, indicating a Fabric Manager is supposed to configure this bridge.
use directed relay Control Reads to check Software Defined Sticky bits
capture manager ID
use directed relay Control Reads and Writes to claim ownership of B1
configure S2 for route to B1
Control Writes to complete B1 configuration
PM / Local Node Services examines message from PFM. Should contain CID of PFM and SFM (if it exists) and capabilities and services available.
Component Local Interrupt to CWM Completion Handler
explicit Control Write Message of introduction from PFM to host on B1
PM / Local Node Services returns capabilities and services it supports, registers with PFM for events and notifications, requests lists of resources and services mapped to this bride.
Local Access Writes initialize Control Write MSG to PFM
explicit Control Write Message to PFM

Figure 4-4: Abdicating Control of a Co-located Bridge

4.4. Securing Boundary Links

Any link that connects two Gen-Z components and meets one or more of the following criteria is considered a boundary link.

- If the link connects components that are not in the same manager domain
- If the link connects components that are not in the same CID namespace (same Fabric UUID)
- If the link has one end connected to a currently un-managed component
- If the link has one end which is connected to an un-powered component, or one which is being held in some form of HW reset
- If the link has one end which is not connected to any component (un-used link)
    - Depending upon the hardware implementation, these latter two cases may appear identical to software

Each component that is detected at the end of a boundary link is classified as one of the following:

- Un-managed, but eligible to be managed by this manager
- Un-managed, but not eligible to be managed by this manager
- Already managed and owned by this manager
- Already managed and owned by a different manager
- Component base class not recognized

Every manager (Primary Manager or Primary Fabric Manager) that configures a component shall evaluate all the link interface structures and peer components attached to them to establish the security state of each link per the
Figure *4-5: Boundary Link Security* Policy, which follows. Not shown in the following figure are policy actions and additional checks to be applied to peer components deemed not to be healthy. Determining component health and policies for link security are topics beyond the scope of this document.

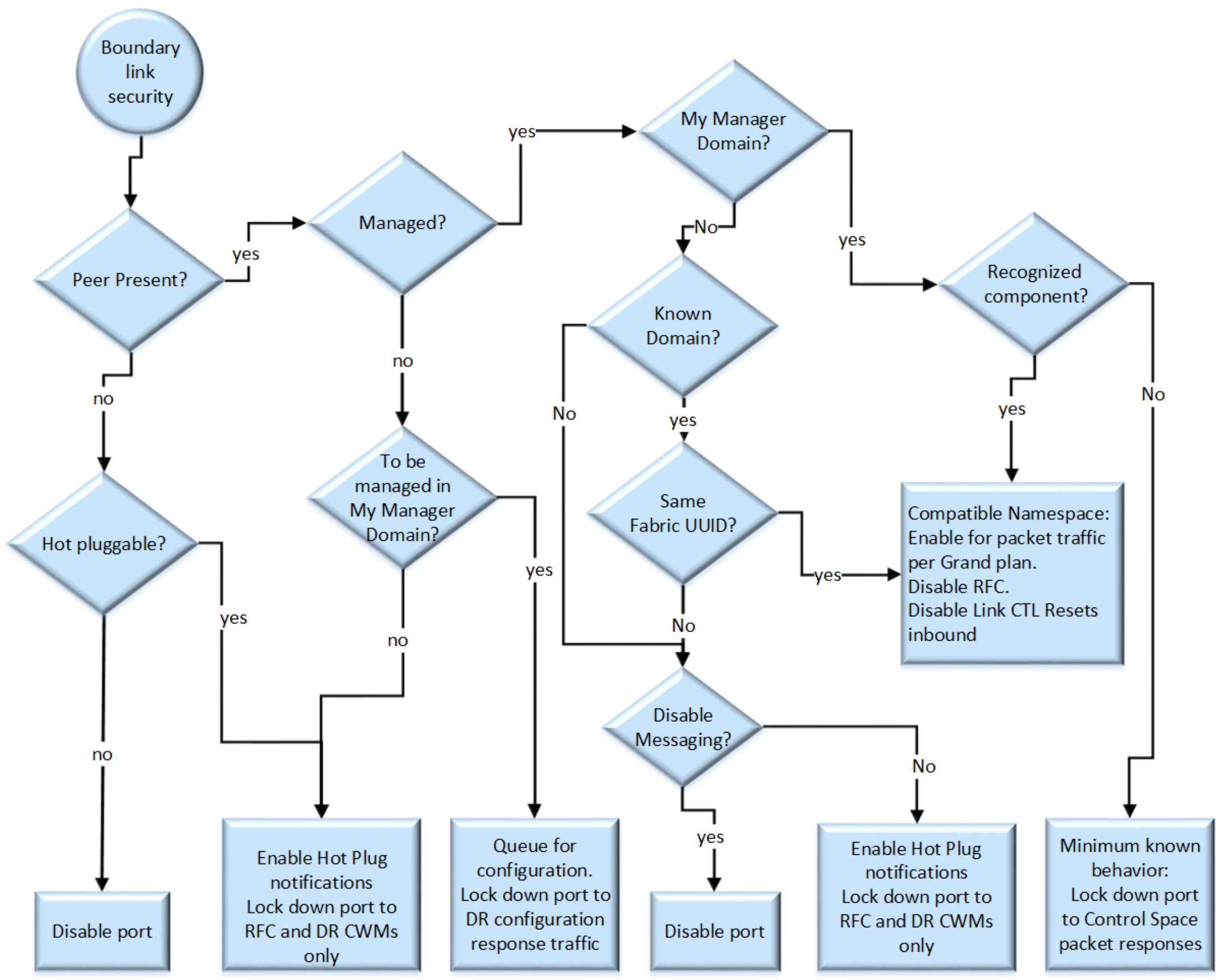

Boundary link security
Peer Present?
yes
no
Managed?
yes
no
My Manager Domain?
No
yes
Known Domain?
No
yes
Recognized component?
yes
No
Hot pluggable?
yes
no
To be managed in My Manager Domain?
yes
no
Same Fabric UUID?
yes
No
Compatible Namespace: Enable for packet traffic per Grand plan. Disable RFC. Disable Link CTL Resets inbound
Disable Messaging?
yes
No
Disable port
Enable Hot Plug notifications Lock down port to RFC and DR CWMs only
Queue for configuration. Lock down port to DR configuration response traffic
Disable port
Enable Hot Plug notifications Lock down port to RFC and DR CWMs only
Minimum known behavior: Lock down port to Control Space packet responses

Figure 4-5: Boundary Link Security Policy

Table 4-2: Boundary Link Security Validation Steps

Validation Step	Explanation
Peer Present?	Link State and or Interface Peer-Attribute fields indicate a peer component is physically present. If a peer component is connected to the link, proceed to determine if the component is un-managed.
No Peer Present: Hot pluggable?	If no physical peer is detected, is the component and/or link interface enabled for surprise hot plug? If no surprise hot plug is allowed on this port (due to policy or HW device or platform abilities), disable the port for maximum security.

Validation Step	Explanation
Managed?	Using tactics outlined in Section *2.3.4 Auto-training links and Link Level Control Packets* determine if the peer component currently has a manager.
No manager: My Manager Domain?	If the peer component is eligible to be managed by this manager based on a pre-populated resource tree per the Grand Plan, or based on Section *4.1 General Manager Domain Exploration Policies,* then proceed with queuing this component for configuration. Enable only responses to directed relay Control Writes and Control Reads to ingress at this port. If the unmanaged component is to be managed by another manager, lock down the port to only directed relay Control Write MSGs and Link CTL Request For Configurations. Set up hot plug alerts.
Managed: My Domain?	If the peer component's CID matches a CID in the manager's domain, confirm this component's MGR-UUID and manager CIDs correspond to this manager. If so, proceed to validating the peer component has a recognized programming model.
Recognized Component?	Manager software shall determine if the peer component is recognized and trusted enough to be allowed to source or sink data space packets through this interface. If not, proceed to set link controls to limit access modes. The decision criteria and the resulting limited access modes are beyond the scope of this document.
Not My Domain: Known Domain?	If the peer component's location in the probing manager's current Grand Plan resource tree and claimed CID(s) match expected details from a list of known adjacent manager domains, use a directed relay Control Write MSG to negotiate with the manager of this peer component, proceed to checking for the same namespace / Fabric UUID If component is not contained in any list of known manager domains, proceed to Disable Messaging check.
Known Manager Domain: **Same Fabric UUID**	If the peer component's manager domain is servicing the same Fabric UUID (same CID namespace), proceed to validating the two domains are compatible. If not, proceed to Disable Messaging check.
Disable Messaging?	If policy or the Grand Plan call for completely isolating this peer component on this link, disable the link completely.

Validation Step	Explanation
	Otherwise, enable this link only for minimal manager to manager communication using directed relay Control Write MSGs as described in Section *4.2 Manager to Manager Communication*.

5. External Standards

5.1. Redfish Fabric Extensions for Gen-Z

The Gen-Z management architecture calls for representing the Gen-Z resources, the Gen-Z fabric, and the services provided by Gen-Z managers as objects within a Redfish base model. As part of an alliance between DMTF and The Gen-Z Consortium, (https://www.dmtf.org/content/dmtf-and-gen-z-consortium-form-alliance) the organizations will collaborate on

- extensions to DMTF's PMCI standards
- an MCTP over Gen-Z binding
- extensions to DMTF's Redfish API to support Gen-Z management

5.1.1. Features required of the Redfish Gen-Z Fabric representation models

The Gen-Z extensions to Redfish are intended to enable the management of port-based Gen-Z fabrics and the associated fabric attached resources. Redfish represents managed entities as a resource tree containing fabric models of the managed objects (switches, switch ports, switch port routes, media controllers, memory, host bridges, bridge ports, etc). The object models contain various properties (EG. type, name, ID, state/status, ports, interfaces) and actions (EG. reset, restart, flush) as well as relationships to other objects (EG. links to enclosures, managers, groups, collections) in the tree.

The current Redfish fabric object models are being extended to handle properties, actions, and relationships that are unique to Gen-Z fabrics. Quite possibly, some of these unique model parameters will be hidden inside opaque Gen-Z specific structures to minimize the impact on more generic fabric modeling and processing.

Management of the modeled objects is performed by manipulating the model structures via Redfish. Redfish is architected as a RESTful interface, is built around the HTTPS protocol and uses JSON for the data format.

The Gen-Z management architecture requires that Gen-Z managers communicate the current inventory and associated state of the components in their manager domains via the Redfish JSON based data models, each containing the appropriate properties, capabilities, actions and relationships as currently defined. Manager to Manager messaging using any transport protocol (Ethernet, Gen-Z, I2C, USB, etc) still uses the Redfish RESTful API and JSON data formats with an appropriate binding. Therefore, when a Gen-Z Composability Manager wishes to reset a specific Fabric Attached Memory module, the Composability Manager can request the Fabric Manager apply the appropriate action to the FAM modules' fabric model using Ethernet, USB, or Gen-Z channels. This request, in turn, will cause the Fabric Manager to issue the appropriate Gen-Z control space packets to the actual component.

The exact requirements, features, and capabilities made available to Gen-Z management software using Redfish are solely defined in the DMTF Redfish specifications, including extensions for Gen-Z fabrics. The Redfish core specification is available at https://www.dmtf.org/standards/redfish. The fabric extensions for Gen-Z will be published therein as they are released.

The following list is representative of the features and capabilities the Gen-Z management stack expects of Redfish:

- all physical components in a single Gen-Z CID Namespace will be represented in the aggregation of all resource trees of each component's manager (typically the Fabric Managers and Primary Managers)
- Every component manager represents all components it manages in the Redfish resource tree it maintains.
- Every component manager hosts the Redfish service root that enables clients to request component state and configuration changes.
- Each manager's Redfish tree may contain objects that represent components that are not managed by this manager. The component data for such objects as stored in this manager's Redfish tree may be incomplete.
- Every component's manager IDs (PFMCID | PFMSID, PMCID, SFMCID | SFMSID) should be discernable by query of the manager's Redfish tree
- Gen-Z subnets should be discernable in a Redfish fabric by query of the manager's tree
- Gen-Z multi-cast Group ID membership should be discernable in the manager's tree by a Redfish query of the manager's Redfish resource tree
- Gen-Z R-Key Domain group membership should be discernable in the Redfish resource tree
- The physical connectivity graph for all components represented in the tree should be discernable from the Redfish resource tree
- Logical resources that represent sub-regions of physical resources should be discernable in the Redfish resource tree
- The Component IDs allocated for use in the manager domain should be discernable in the Redfish resource tree
- Redfish 'manager' objects should contain the manager's hostname.
- Redfish authorization techniques should enable the fabric admin to create at least one a priori client (the Composability Manager) in a manager domain resource tree which is authorized to affect changes to the fabric components therein.

5.1.2. Required Redfish Support in Gen-Z Managers

To enable optimizations in multi-manager use cases, Gen-Z Fabric Managers and Primary Managers shall maintain a local Redfish service root which they use to track the components in their manager domain. The local resource root structure may or may not contain all the global relationships tracked in the top level Grand Plan.

However, the local FM and PM resource root structure shall contain the following:

- a Redfish fabric model for each component in the manager's domain
- a resource tree that reflects the logical topology of the manager's domain as viewed with the manager as the service root
- a Redfish placeholder model for each peer component on the other side of this manager's boundary links
    - Placeholder model data may be incomplete
    - Manager IDs (hostname, GCIDs, boundary network Gen-Z IP address if known) of placeholder components are stored in the appropriate Redfish data object within this managers' resource tree.
- one or more Redfish objects to define the allowed global component IDs (GCIDs) allocated to Gen-Z devices within this manager's domain
- one or more Redfish objects to define the necessary Grand Plan details, including, but not limited to:
    - The intended manager role(s) to be provided by the Redfish service manager (the manager of the domain)
    - The component authentication policies to be instituted by the manager
    - The fall back policies that dictate what a Redfish resource manager does when it must deviate from the a priori (local) grand plan.
    - The Fabric UUID
    - Any specific exceptions to the Rules of Exploration that apply to this fabric and/or specific managers within it
    - Any specific security policies to be enforced at boundary links of the various types, per *Section 4.4 Securing Boundary Links.*

Further, if a manager entity is acting as a Secondary Fabric Manager, this Secondary Fabric Manager shall:

- Maintain a separate Redfish service root that contains all of the objects stated above that the Primary Fabric Manager maintains.
- a Secondary Fabric Manager's models and data shall be consistent with the Primary Fabric Managers models

The Fabric Managers and Primary Managers are responsible for creating and maintaining their Manager domains' Redfish resource trees. Redfish clients of these managers' services may request changes to Gen-Z fabric components or fabric state if properly authorized. Thus, the Composability Manager generally must request the Fabric Managers and Primary Managers proxy changes to the Gen-Z hardware. The associated changes to the Redfish resource trees are also made via these proxies.

To maintain a consistent view of all Gen-Z fabric resources and the global CID Namespace under which they are managed, **the Composability Manager** (or that entity which maintains the global Grand Plan for the entire Gen-Z fabric) may create and maintain an aggregation of all resource trees hosted by the Fabric and Primary Managers in its global CID namespace. The Composability Manager is a Redfish client of these Fabric and Primary Managers, so it may use Redfish queries, commands, aggregator services, and event services to maintain a valid global view of the fabric resources, and to make changes in component configurations and states to change the global functionality of fabric.

This specification does not define methods for maintaining

- the global Grand Plan,

- the global aggregation of Redfish fabric resource trees,
- any redundant trees hosted by redundant managers, and
- synchronization of the actual state of Gen-Z hardware across all the different representations of Gen-Z resources.

5.2. MCTP Over Gen-Z Using Control Write Messages

As stated in *Section 5.1 Redfish Fabric Extensions for Gen-Z*, the MCTP over Gen-Z binding will be defined by DMTF. MCTP is a protocol designed to support communications between different intelligent hardware components that make up a platform management subsystem. MCTP protocols are not useful for manipulating configuration and state of standard Gen-Z components. However, the Gen-Z Core Specification anticipates at least two significant tasks wherein a Gen-Z manager would desire to use industry standard MCTP messages rather than define yet another set of messages to accomplish them:

- Per the Gen-Z Core Specification (Section 16.6 Component Authentication) Component Authentication on the Gen-Z fabric is to be performed using the MCTP objects and procedures defined in an upcoming *DMTF SPDM Specification*.
- MCTP protocols are a popular mechanism for transferring standardized component control operations over various transport mechanisms.
    - For example, MCTP over PCIe is a DMTF binding that allows a component manager to direct common fan control commands to a PCIe endpoint which would unpack these commands and forward them via some other transport, (EG. I^{2}C or IPMB) to the target fan device.

  The MCTP over Gen-Z binding will enable similar control paradigms for non-Gen-Z components that are connected to a Gen-Z endpoint. This binding is to be defined in an upcoming *DMTF MCTP over Gen-Z Specification.*

  Developer Note: *At the time of this writing, the DMTF SPDM Specification covering security session establishment and the DMTF MCTP over Gen-Z Specification are under development. Once they are finalized, this section will be updated to reflect any additional details unique to Gen-Z that need to be specified. This same note appears in the Gen-Z Core Specification.*

The Gen-Z Core Specification (Section 8.7) declares unreliable Control Write Message packets are the primary mechanism for encapsulating MCTP messages over Gen-Z fabrics. When sending MCTP messages in the payload of a Control Write Message, the sender uses RSPCTXID == 1 to indicate the payload formatting.

Developer Note: *Control Write Messages come with certain restrictions when tunneling MCTP messages*

- *Components that support in-band management are required to support multi-packet Control Write Messages with total message contents of up to 1536 Bytes.*
    - *MCTP message formats enable multiple packet messages. Software must manage multi-packet MCTP messages as defined in the payload binding and cannot rely on Gen-Z multi-packet message facilities for MCTP messages that may be longer than 1536 bytes.*

- *The Gen-Z requestor may need to use the directed relay form of unreliable Control Write Messages to send an MCTP payload across a boundary link.*
- *The target of the MCTP message may need to send a reply message. The original MCTP message does not contain the Gen-Z CID of the original Requestor. The MCTP over Gen-Z binding needs to comprehend this limitation.* 5.1

5.3. Manager to Manager Messages

As outlined in Section *4.2, Manager to Manager Communication,* Gen-Z managers need to be able to communicate with each other using Gen-Z Control Write MSG as the basic in-band mechanism. The general format of the Control Write Message is found in the Gen-Z Core Specification (Section 6.10.2). In that packet there is only a simple payload field defined. Here, we will define the payload field format for Manager to Manager Messages sent via the Control Write MSG packets.

The following definitions for Gen-Z Manager to Manager messages encoded into the Control Write MSG packet payload are very preliminary and subject to change greatly between this draft revision and the final version 1.0 candidate. The reader is advised to retrieve any updates to this document issued between this version and the final candidate, and to provide feedback.

Section 2.3.1 of this documents requires Gen-Z components which exchange Control Write Messages to implement a minimum message payload size of 1536 bytes. This requirement is made to enable the direct encapsulation of the standard 1500 byte MTU message payload size of Ethernet frames.

5.3.1. The Gen-Z Control Write MSG Network Stack

Figure 5-1 illustrates the effective networking stack this specification assumes will be used by various management software entities such as the Primary Managers, Fabric Managers, and Composability Managers, when they are communicating with each other across the Gen-Z fabric. Private IPv4 addressing is assumed for the sake of illustrations and examples, but it is not mandatory. See also *Section 5.3.2 The Gen-Z Boundary Link Private IPv4 Network.*

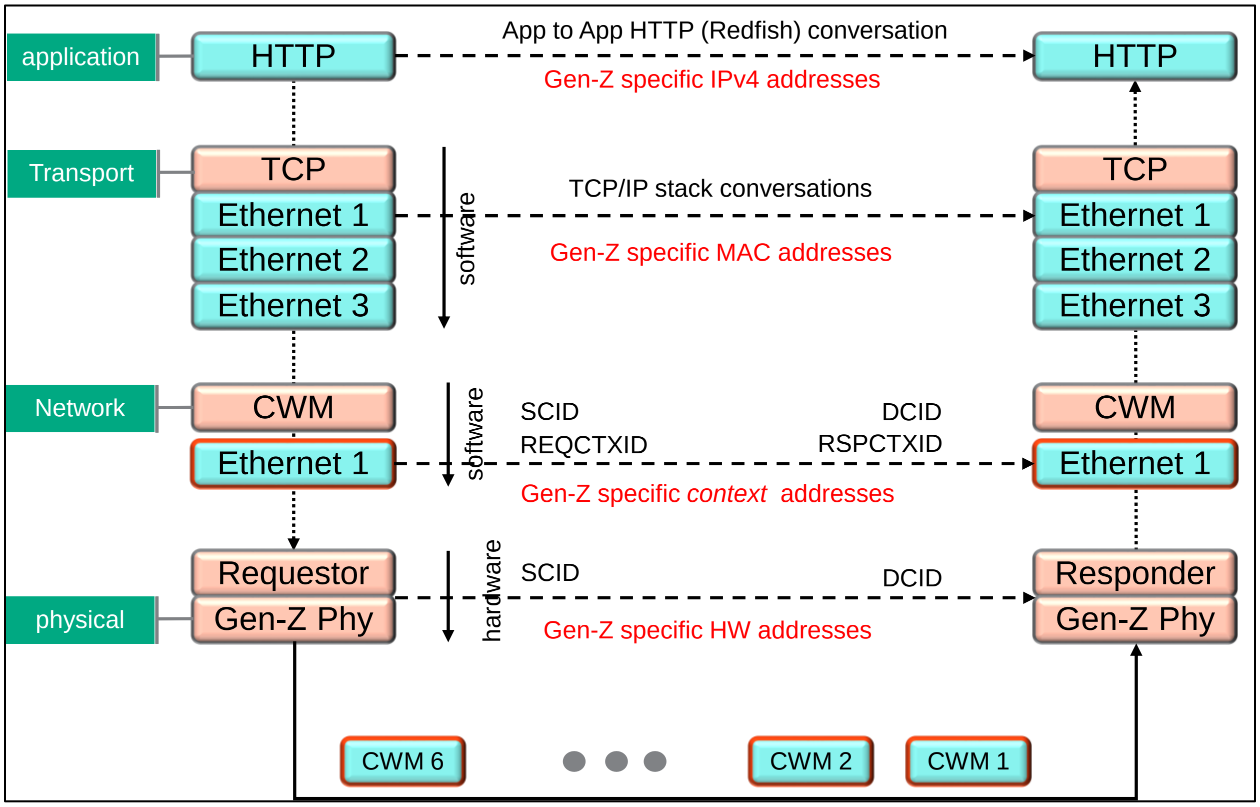


Figure 5-1 Gen-Z Management Messaging layers Using TCP/IP

Because Gen-Z managers use Redfish schema and data models, which in turn are manipulated using HTTP or HTTPS messaging protocols, a large majority of messages between managers can be sent via a form of HTTP encapsulated into Control Write Messages. Even messages that are not focused on the Redfish representations of a manager's domain likely have a corresponding representation which can be found in MPI, IPC, or RPC standards which will have an HTTP protocol equivalent.

Further, most HTTP messages are actually sent over TCP/IP (Ethernet) transports, so herein we define a simple Ethernet payload format that allows most managers to exchange HTTP and Redfish messages using the Gen-Z fabric as the medium and the Control Write MSG packet as the protocol on the fabric.

Ethernet layer 2 frames are 64 bytes to 1526 bytes in length, including the Ethernet header, two optional 4 byte 802.1Q tags, the Ethernet payload, and an internal 4 byte CRC. These encapsulate nicely into the Gen-Z Control Write MSG payloads of 64 - 1536 bytes supported by Gen-Z manager devices per *Section 2.3.1 Assumptions and Requirements*.

Error! Reference source not found. The typical networking layers for Gen-Z Manager Messaging are roken down into the following four groups:

- Application level communications between managers using HTTP / Redfish and other standard messaging APIs to communicate.
    - Manager applications identify each other at this level via a public or private IP address.
    - When Gen-Z fabric is the underlying transport, Managers may use any of the Private IP networks defined for IPv4 or IPv6, or any proprietary IP address formats that are compatible with these standards.

- Gen-Z Transport TCP/IP stacks that provide reliable transport of HTTP style messages by breaking them down into multiple Ethernet 'frames'. This layer requests re-transmission of missing frames to make the HTTP message whole.
    - TCP/IP stacks address each other using local or global MAC addresses.
    - When Gen-Z fabric is the underlying physical transport, Managers shall use the Gen-Z specific local MAC addressing formats defined in ***Error! Reference source not found.***.
- Gen-Z Network stacks that set up management context buffers, fill them with message contents (using Ethernet frames or other formats), and transfer them to the target device's receiving buffer using Control Write Messages.
    - Gen-Z management networking layer identifies the source and target buffers via Gen-Z specific Context IDs concatenated with other fields from the Control Write MSG headers.
    - The REQCTXID and RSPCTXID values of '0' are reserved for Gen-Z manager use, and managers shall use these '0' values as the default contexts until the manager pair agrees to use different values.
        - The mechanism to declare an alternative Context ID for management Requestors and Responders to use is outside the scope of this document.
    - When the Gen-Z Network stack buffers Ethernet frames for transmission, it shall inspect the Destination Gen-Z MAC address which dictates it select either the explicitly addressed Control Write MSG packet, or the directed relay unreliable Control Write MSG packet to transmit the buffer contents on the physical Gen-Z fabric.

- Gen-Z Requestor and Responder HW physical transport layers which parse the context buffer message contents and pack them into one or more Control Write MSG packets, addressed strictly by Gen-Z global Component IDs.

Management software may wish to make use of the Gen-Z *reliable* Control Write MSG messaging protocol directly between application layers of managers when the appropriate messaging capabilities are supported by the Gen-Z hardware. For example, Gen-Z managers may exchange custom messages as ASCII encoded character strings by calling directly into the Gen-Z networking layer and requesting the appropriate messaging operation be launched using the Software Defined CWMSG Payload Format and referencing the appropriate Requestor Context ID (REQCTXID) and Responder Context ID (RSPCTXID) designations.

- Management software is responsible to ensure that the full custom message is not larger than the maximum reliable Control Write Message length
- Management software is responsible for handling the rare but possible failure modes associated with the reliable Control Write MSG protocols.
- Management software is responsible for handling the issues associated with message and packet re-ordering
- The process for negotiating the use of Software Defined Payload Format messages and the choice non-zero REQCTXID and RSPCTXID values is beyond the scope of this specification.

The Gen-Z HTTP Payload Format Type (optional) has been defined to enable management software to pre-define RSPCTXID values with the Redfish (HTTP subset) message payloads. See Section *5.3.4 HTTP or SW Defined Format Manager Messages* for further details.

5.3.2. The Gen-Z Boundary Link Private IPv4 Network

Section 4.4 Securing Boundary Links dictates that when two manager domains are connected but both are yet not validated to be compliant to the same GCID Namespace, the managers need to communicate using only the directed relay, unreliable Control Write MSG protocols across the boundary link. In *Section* ***Error! Reference source not found. Error! Reference source not found.*** it was stated that the fault IP addressing scheme is to be IPv4 when accessing another manager via TCP/IP standards. These requirements enable managers from different vendors to quickly create application level connections using ubiquitous networking stacks. Such stacks emit standard Ethernet formatted Control Write Messages to the hardware transport layers.

To enable a generic TCP/IP stack to work using a Gen-Z boundary link as the physical layer, there must be network addressing system exposed to the stack. To that end, this specification requires each boundary manager to be capable of creating a private IPv4 network on top of the boundary link as the physical layer.

These are the requirements of such a private network:

- Private Network shall be 192.168.0.0/24 format
- The creating manager shall be the gateway NAT router between its boundary link private network and any other IP networks (standard Internet, private subnet, other Gen-Z IP subnets)
- The creating manager shall have a host / domain name valid on all networks it routes among
- The creating manager shall have valid network IP addresses on all networks it routes among
- The creating manager shall be the DHCP, DNS, and reverse DNS services for its boundary link private network
- The creating manager shall dissolve the boundary link private network and recover the private network subnet number once the two manager domains have been merged into the same fabric GCID Namespace and explicitly addressed Gen-Z packets can be transported across the boundary link.

Section ***Error! Reference source not found. Error! Reference source not found.*** defines how the two nagers negotiate which one is to create and manage the boundary link private network.

5.3.2.1. Negotiation of Boundary Link Network Owner

Section *5.3.2 The Gen-Z Boundary Link Private IPv4 Network* indicated one of the first things two boundary managers do after authenticating each other is to create a boundary link private IPv4 network so the managers can exchange Redfish protocol messages through a standard TCP/IP software stack.

The two managers must first negotiate which of them will be the creator of the boundary link network. The creating manager is referred to as the 'owner' or 'owning manager' of private network. The negotiation process makes use of these fields from the Get Mgr Domain Info message payload:

- Requestor and Responder CM Status fields
  - The CM Status field describes the general crawl out state of a manager domain. A boundary link network is not required until at least one of the boundary manager domains has connected with its Composability Manager (be it co-located with the domain manager or external to it)

  - The crawl out states tracked in the CM Status field are roughly characterized as
    - Standalone manager domain not yet completely configured
    - Standalone manager domain fully configured
    - Fabric member manager domain without a CM validated configuration
    - Fabric member manager domain with a CM validated configuration

- Requestor and Responder Subnets Available fields
  - Managers indicate the number of subnet values they have available to assign to the boundary link private network
- Requestor and Responder RND number fields
  - Managers generate and publish a Random number to use as a tie breaker when negotiating for ownership of the private network
- Private Subnet Request, Private Subnet Response, and Private Subnet Owner fields
  - These are the fields through which the Managers indicate their intentions to determine a network owner and tender their decision

5.3.2.1.1. Manager Priority for Ownership

The manager priority for ownership is determined using the following tie-breaker rules, given in order of evaluation:

- If a Manager does not want to be the private network owner the other Manager has priority and is declared the winner and the owner. If neither manager refuses the ownership role, then
- The Manager with the highest CM State has priority and is declared the 'winner' and thus the owner. If the two managers tie on CM State, then
- The Manager with fewest available subnet numbers has priority and is the owner. If the two managers tie on available subnet numbers, then
- The Manager with lowest Random Number included in original exchanges has priority and is the owner. If the two managers tie on their Random Numbers, the negotiation process is repeated.
  - If both managers issued requests for negotiation for ownership, each may randomly delay issuing a new request.
- *If both Managers refuse the role of owner of the private network, either manager may reconsider their choice and send a new 'request for ownership negotiation'. However, neither is required to do so.*

The negotiation process is as follows:

1. One manager chooses to initiate the negotiation for ownership, and asserts the Private Subnet Request code in a Get Mgr Domain Info Request message.
   a. The initiating manager includes its CM Status value in the Requestor's CM Status field
   b. The initiating manager includes the number of available subnet values it has available at this time in the Requestor Subnets Available field
   c. The initiating manager includes a random number (from the applicable range) in the Requestor RND field.
   d. The initiating manager sets the appropriate bit of the request's MSG Status field to indicate private networking request and info fields of the message contain valid data.
   e. The initiating manager sets the request message's Private Subnet Owner field to 'no ownership resolution'.

   f. The initiating manager issues the Get Mgr Domain Info Request message via a directed relay, unreliable Control Write MSG packet across the boundary link to the peer manager.
      i. The manager may include other relevant information about its manager domain in the same request message, but until a subnet owner is chosen, no other networking information, requests, or responses are allowed.
   g. The initiating manager sets up its unreliable message timeout counter and waits for the peer manager to send a Get Mgr Domain Info Response message in reply.
      i. If the timeout counter pops, and the peer manager has not sent a reply or its own 'negotiation for ownership' Get Mgr Domain Info Request message, this initiating manager may retry this request message.
      ii. The policies that establish the timeout wait duration and the min and max retry values are beyond the scope of this specification.
2. The peer manager receives the Get Mgr Domain Info Request message and decodes the payload contains the initiating manager's request to negotiate the private network owner, and constructs its Get Mgr Domain Info Response message.
   a. The recipient manager includes its CM Status value in the Responder's CM Status field
   b. The recipient manager includes the number of available subnet values it has available at this time in the Responder Subnets Available field
   c. The recipient manager includes a random number (from the applicable range) in the Responder RND field.
      i. If the recipient manager had also issued its own request message to initiate ownership negotiations, it shall not issue additional requests until it has received a response to its first request.
      ii. If the recipient manager has already issued its own 'request for ownership negotiation' request message, the recipient manager shall use the same values from its request as the CM Status, Responder Subnets Available, and Responder RND values in this response.
         1. Both managers will see the same details from the other manager, whether they are included in a request message or its response.
      iii. If the recipient manager has no pre-chosen value, the recipient manager shall choose its value for Responder RND field before it evaluates the resulting manager priorities for ownership.
      iv. The recipient manager will not change the value it uses for the Responder RND field unless the negotiation process is initiated again because of a complete tie in the evaluation process underway.
   d. The recipient manager proceeds to evaluate the two managers' priorities per *Section* ***Error! Reference source not found. Error! Reference source not found.*** and fills in the propriate Private Subnet Owner field accordingly.
      i. If the recipient manager cannot or will not be the owner of the private network, it shall still fill in the required response fields
   e. The recipient manager issues the Get Mgr Domain Info Response message to the original requestor using a directed relay, unreliable Control Write MSG packet.

3. The initiating manager receives the anticipated Get Mgr Domain Info Response message and decodes the payload contents.
   a. The initiating manager extracts the responding manager's CM State, Responder Subnets Available, Responder RND, and Private Subnet Response values.

   b. The initiating manager runs its own evaluation of the two managers' priority tie breaker values, per *Section* ***Error! Reference source not found. Error! Reference source not und.*** and reaches its own conclusions about who the Private Subnet Owner should be.
   c. If the responding manager has sent its own 'request for ownership negotiation', this initiating manager may eventually see this request.
      i. This initiating manager shall respond to the other manager's request with the same data it included in its own request, and with the same conclusion on ownership.

4. If the responding manager conclusions agree with the initiating manager's conclusions:
   a. If the requesting manager is the chosen owner, it may begin negotiating the boundary link subnet number.
   b. If the responding manager is the chosen owner, the requesting manager simply waits for the other manager to begin negotiating the boundary link subnet number
   c. If there is no clear owner as a result, both managers shall proceed per *Section* ***Error! eference source not found. Error! Reference source not found.***.

5. If the two managers did not reach the same conclusion:
   a. If the responding manager chose the requesting manager, the requesting manager may request new negotiations
   b. If the responding manager chose itself as the owner, the requesting manager awaits a new request from the other manager, (the contested owner.) Unless the new request is a request to re-start negotiations for ownership, the requesting manager shall respond to any such request messages with Private Subnet Owner == 0x3, owner conflict.
   c. If the responding manager did not choose an owner, but the initiating manager did, the initiating manager shall create a new request to start private network ownership negotiations.

5.3.2.2. Arbitration of Boundary Link Subnet Numbers

Section 5.3.2 The Gen-Z Boundary Link Private IPv4 Network defines the private networks each manager creates to bridge a boundary link. *Table 5-4 Mgr Domain Info Message Payload Field Definitions* describes where within the Mgr Domain Info Message each boundary owner proposes a unique subnet number, tracks the status of its proposal, and where the receiving manager deposits its response to the owning manager's proposal. This section describes the process to be used to choose the unique subnet number for each manager's private subnet, and the requirements to be met when selecting them.

Each private boundary link network shall:

- Have a subnet number unique from all other boundary link subnets within the manager domain.
  - A domain manager may have several boundary links within its domain.
  - A domain manager may have more than one boundary link associated with a single boundary component.
  - Management software is responsible for ensuring any boundary component's manager has enough available subnet numbers to enable a unique value to be chosen for all boundary link networks.

- Have a subnet number unique from that of the other boundary link manager on the peer side of the link
- Have at least one device IP address available to assign to the manager on the other side of the boundary link

The process to negotiate the boundary link subnet numbers is as follows:

- Owning manager issues a Get Mgr Domain Info Request Message across the boundary link using the directed relay, unreliable Control Write MSG packet with the following fields initialized as follows:
    - Message Status indicates that the Requesting Manager's manager details and networking proposal fields are valid in this request.
    - Private Subnet Request indicates this message contains a proposal for the Requestor's subnet value.
    - Private Network Subnet set to a valid, non-zero number
    - Private Subnet Owner indicates this manager owns the private network
- Receiving manager issues a Get Mgr Domain Info Response Message with the appropriate original request values mirrored in the response along with the receiving manager's reply:
    - Private Subnet Request and Private Network Subnet values are reflected from the original request
    - Private Subnet Owner now indicates this manager sending the response is not the owner
    - Private Subnet Response indicates if the recipient manager accepts this subnet value or not.
- Owning manager evaluates the response message and acts accordingly
    - If the subnet value proposed by the owner was accepted, owner issues confirmation
    - If the subnet value was not accepted, the owner tries another one.
    - If the subnet value was not accepted and the owner has none left to try, it may recover subnet values no longer in use because the associated boundary link was opened to explicitly addressed Gen-Z packets.
- Once the owning manager has negotiated an acceptable subnet number with the peer manager, it may place its boundary link private network into service and notify the peer manager that it has done so with the appropriate Private Subnet Request value in another Get Mgr Domain Info Request.

- To correct any errors in the subnet value negotiation process, or to finish the tear down of the boundary link private network after it has served its purpose, the owning manager may use the 'Revoke Subnet value' code in a Private Subnet Request to terminate use of a given subnet number.

5.3.2.3. Requesting Boundary Link Gen-Z IP Addresses

Once the private network owner is selected and a subnet number is agreed upon, the owning manager can place the boundary link network into service. As soon as the non-owning manager is alerted that the network is active, it will need to request its own Gen-Z IP Address on the private network.

The non-owning manager obtains one of the available Gen-Z IP Addresses from the owning manager by requesting one using these fields of the Get Mgr Domain Info Request message:

- GET GZIPA field contains the 4-bit subnet device ID of a Gen-Z IP Address 196.168.0.0 / 24 private subnet
- GET GZIPA Status contains the 4-bit request or 4-bit response that allows the two managers to exchange requests for IPAs, offers of IPAs, confirmations of IPAs, or revocations of IPAs.

- See *Section 5.3.6 Get Mgr Domain Info Messages* for complete details.

The process for obtaining a Gen-Z IP Address on the boundary link network is very similar to the DHCP process of Discovery, Offer, Request, and Acknowledgment (DORA). Since the boundary link network has no broadcast mechanism, and only to endpoints at initial start-up, the private network's equivalent to DORA is much simpler.

- The non-owner manager launches a Get Mgr Domain Request message directly to the owning manager, and includes:
  - The GET GZIPA status value which requests the owner return a valid Gen-Z IP Address in the response.
    - The requesting manager may indicate the IP Address it would prefer be assigned by including it in the GET GZIPA field of the request message, and using the appropriate GET GZIPA Status request code.

- The owning manager sends its Get Mgr Domain Response message with the following included:
  - The GET GZIPA 4-bit device ID being offered to the requesting manager
  - The associated GET GZIPA Status response code indicating the results of the request
  - The owner's Gen-Z IP Address device ID shall be included in the Owning Mgr GZIPA field of the response

- The initiating manager (the non-owner) will examine the response packet and proceed accordingly:
  - If the owner returned the GZIPA value requested, or any value that is acceptable, the non-owner shall issue another Get Mgr Domain Request and inform the owner that the non-owner is accepting the offered IP value.
  - If the owner failed to return a valid device IP value, the initiator needs to assess the reason for failure.
    - The owner should not run out of available IP addresses if it has not issued them to the non-owner.
    - Since there is no 'lease duration' limit in this simplified DORA scheme, possibly the non-owner has requested and received several IP addresses in the past and has not released them.
      - The non-owner can offer to surrender various values whether it believes it was granted them earlier or not.
      - The owner will reject the offers to release IP addresses it believes the non-owner was never granted.
    - .

- Once the non-owner manager has a valid Gen-Z IP Address on the boundary link private network, it may enable management traffic between the two managers to use HTTP or Redfish protocols through a standard TCP/IP stack. The typical TCP/IP stack will convert these protocol messages to Ethernet frame payloads sent over the Gen-Z boundary link transport via directed relay, unreliable Control Write MSG packets.

- If no free IP addresses can be found on the boundary link private network, or if the network cannot be started for some other reason, the managers may have to rely on the raw HTTP and Software defined packet formats to transfer high level messaging protocols running outside the standard TCP/IP stack. Such messaging stack APIs and setup flows are outside the scope of this specification

5.3.3. Mandatory Control Write MSG Header

The first 8 bytes of the Control Write MSG message payload is reserved as the mandatory header that defines the format of the payload.

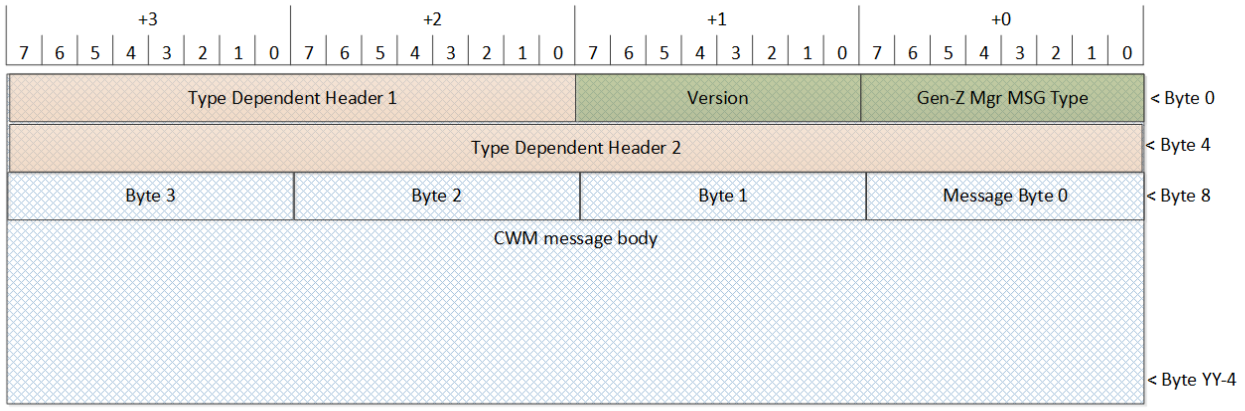


Figure 5-2 Control Write MSG Payload Formatting

The Mandatory Control Write MSG header contents shall be organized within the sending and receiving message buffers (designated partially by Control Write MSG header fields REQCTXID and RSPCTXID, respectively) as shown in ***Error! Reference source not found.***.

- Byte 0 of the context buffer corresponds to byte 0 of the *first* Control Write MSG packet payload as described in the Gen-Z Core Specification v1.1 (Section 6.10.2.7).
- Byte 0 of the context buffer thus contains the Gen-Z Manager Message Type field

The following table describes the required fields of this payload header.

Table 5-1 Control Write MSG Payload Field Definitions

Field Name	Size	Description
Gen-Z Mgr MSG Type	8 bits	Gen-Z Manager Message Type defines how the rest of the header and remaining payload contents are formatted and processed. Refer to *Table 5-2 Manager Message Type Encodings* for details.

Version	8 bits	Declares the version of the Control Write MSG Payload Header definition used to format this packet. Current revision # for this field is 0x01 Version 0x0 is Reserved.
Type Dependent Header 1	16 bits	Field interpretation defined by Gen-Z Mgr Msg Type encoding Some message encodings may use this field as additional message payload
Type Dependent Header 2	32 bits	Field interpretation defined by Mgr Msg Type encoding Some message encodings may use this field as additional message payload
CWM message body	N * 32bits	The remaining Control Write MSG payload field contains the body of the message declared in the header fields. The Control Write MSG packet format is defined to be a multiple of 4 bytes in total length, so the payload field is padded with zero to three bytes of zeros by the hardware. These pad bytes are not passed to the RSPCTXID buffer holding the message.

The Gen-Z Manager Message Type encodings are defined in *Table 5-2 Manager Message Type Encodings.* Gen-Z managers shall support all defined message types unless the type is specifically declared to be optional.

Table 5-2 Manager Message Type Encodings

Message Type (8 bit unsigned integer)	Encoding	Description
Software Defined Payload Format (optional)	0x00	Software Defined CWMSG payload format: Management software may choose to pass custom formatted Control Write MSGs using appropriate REQCTXID / RSPCTXID pairs, wherein the RSPCTXID shall be non-zero. Software associated with the packet's RSPCTXID will interpret the format of the message Type Dependent Header fields and remaining payload contents. If RSPCTXID == 0, this Message Type encoding shall be Reserved. Software may use this Message Type to bypass the TCP/IP stack and rely on Gen-Z reliable Control Write MSG packets to connect application layers. Support of this Manager Message Type is optional.

		See *Section 5.3.4 HTTP or SW Defined Format Manager Messages*
Ethernet Format	0x01	Ethernet Format in CWMSG payload: Management software may encapsulate standard Ethernet frames within Control Write Messages. The Type Dependent Header fields and remaining payload contents are formatted as described in Section *5.3.5 Ethernet Formatted Manager Messages.*
HTTP Payload Format **(optional)**	0x02	HTTP CWMSG payload format: Management software may choose to pass HTTP formatted Control Write MSGs using appropriate REQCTXID / RSPCTXID pairs, wherein the RSPCTXID shall be non-zero. Software associated with the packet's RSPCTXID will interpret the format of the message Type Dependent Header fields and remaining payload contents as HTTP and route the completed message to the named URI service. If RSPCTXID == 0, this Message Type encoding shall be Reserved. Software may use this Message Type to bypass the TCP/IP stack and rely on Gen-Z reliable messaging packets to connect application layers. Support of this Manager Message Type is optional. See *Section 5.3.4 HTTP or SW Defined Format Manager Messages*
	0x03 – 0x07	Reserved
Get Mgr Domain Info **Request**	0x08	Get Manager domain Information Request messages are the initial requests and responses sent between managers at opposite ends of a boundary link. The information exchanged enables private IP networks to be instantiated across the link so subsequent manager to manager messaging can utilize Redfish protocols. See *Section 5.3.6 Get Mgr Domain Info Messages*
Get Mgr Domain Info **Response**	0x09	See *Section 5.3.6 Get Mgr Domain Info Messages* The responses to Get Mgr Domain Info request messages are defined within the subsection that defines the request.
	0x0a – 0x7f	Reserved for Gen-Z Consortium definitions
	0x80 – 0xff	Vendor Defined messages

5.3.4. HTTP or SW Defined Format Manager Messages

This specification defines the following encapsulation of an HTTP or a Software Defined Message into the Control Write Message payload.

The HTTP or Software Defined Message contents shall be organized within the sending and receiving message buffers (designated partially by Control Write MSG header fields REQCTXID and RSPCTXID, respectively) as shown in

1. Byte 0 of the context buffer corresponds to byte 0 of the *first* Control Write MSG packet payload
   - Byte 0 of the context buffer thus contains the Gen-Z Manager Message Type field
2. Byte 2 of the context buffer contains byte 0 of the encapsulated message contents

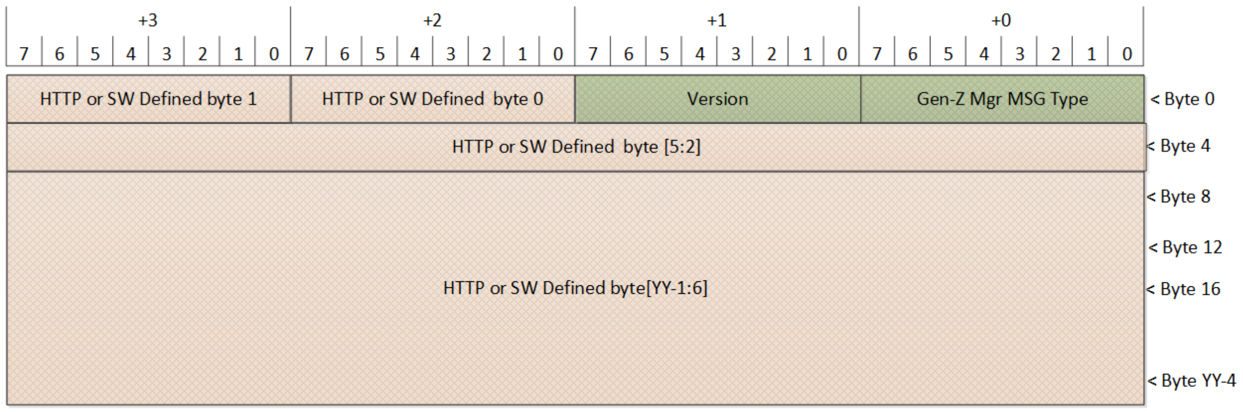

+3
+2
+1
+0
7 6 5 4 3 2 1 0
HTTP or SW Defined byte 1
HTTP or SW Defined byte 0
Version
Gen-Z Mgr MSG Type
HTTP or SW Defined byte [5:2]
HTTP or SW Defined byte[YY-1:6]
< Byte 0
< Byte 4
< Byte 8
< Byte 12
< Byte 16
< Byte YY-4

Figure 5-3 HTTP or SW Defined Payload Format within Context Buffer

5.3.5. Ethernet Formatted Manager Messages

This specification defines the following encapsulation of Ethernet II and IEEE 802.3 Ethernet frames for use with the Ethernet Format Manager Message Type:

The Ethernet frame contents shall be organized within the sending and receiving message buffers (designated partially by Control Write MSG header fields REQCTXID and RSPCTXID, respectively) as shown in *Figure 5-4 MGR MSG Ethernet Frame Format within Context Buffer.*

3. Byte 0 of the context buffer corresponds to byte 0 of the *first* Control Write MSG packet payload
   - Byte 0 of the context buffer thus contains the Gen-Z Manager Message Type field
4. Byte 2 of the context buffer contains byte 0 of the Ethernet Frame contents

5. If the Ethernet *frame size* exceeds 254 bytes, more than one Control Write MSG *packet* will be required to transmit the complete message. The CWMSG packet header contains the necessary information to re-assemble the completed message
   - Only the first packet will contain the Manager Message Type and Version header shown
   - The first packet will use WOFF == 0 in the Gen-Z Control Write MSG packet header

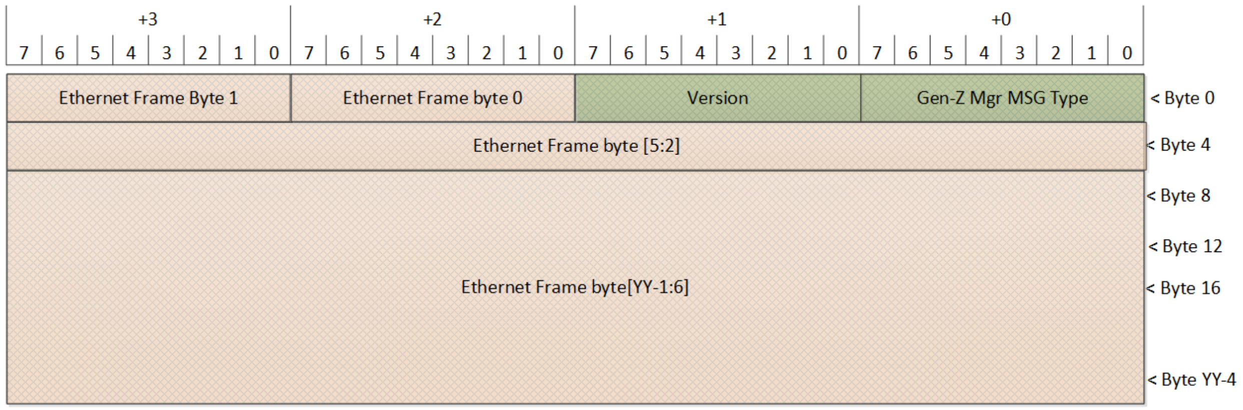


Figure 5-4 MGR MSG Ethernet Frame Format within Context Buffer

The normal Ethernet message header has a 48-bit MAC address for the source, and a 48-bit MAC address for the destination. Since these messages are transported over Gen-Z fabrics via the Control Write MSG packet, the globally unique MAC addresses typically used on internet fabrics are inappropriate.

Gen-Z managers shall use a 'locally defined' MAC address as defined below.

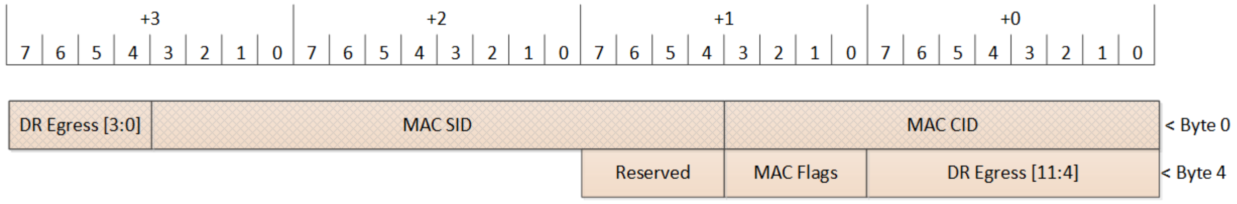


Figure 5-5 Gen-Z Local MAC Address Format

Table 5-3 Gen-Z Local MAC Field Definitions

Field Name	Size & Type	Description

MAC CID	12 bits	The Component ID (CID) of the manager which is the source or destination of the Ethernet message
MAC SID	16 bits	If MAC flags Bit 2 == ‘1b’, this is the Subnet ID (SID) of the entity associated with the MAC CID address Otherwise this field is Reserved
DR Egress	12 bits	If a Destination MAC, If MAC flags Bit 3 == ‘1b’, this is the Egress port number to which the directed relay Control Write MSG should be relayed by the component whose GCID is listed as the destination MAC (CID | SID). If this is a Source MAC, if MAC flags Bit 3 == ‘1b’, this is the ingress port from which the directed relay Control Write MSG was received by the component whose GCID is listed as the source MAC (CID | SID). If MAC flags bit 3 == ‘0b’, this field is Reserved.
MAC flags	4 bits	The MAC address flags are decoded as follows when using Control Write MSG packets with Ethernet payload: Bit 0: MC: multicast bit, shall be ‘0b’. Bit 1: L: locally defined MAC, shall be ‘1b’ Bit 2: GC: global CIDs in use, Set to ‘1b’ if MAC SID is valid Cleared to ‘0b’ otherwise Bit 3: DR: Directed Relay Control Write MSG packets needed When set to ‘1b’ in a Destination MAC address, the directed relay, unreliable Control Write MSG packet shall be used by the sender.
Reserved	4 bits	Reserved for future use

5.3.6. Get Mgr Domain Info Messages

This section describes both the Get Mgr Domain Info Request and the closely related Get Mgr Domain Info Response messages.

This manager to manager message is expected to be delivered mainly using the directed relay, unreliable Control Write MSG packet. Therefore this message request and message response versions are crafted to fit within the 256 byte payload of a single Control Write MSG for convenience. Further the Request version of this message may contain the associated values of the requesting manager's Manager Domain, which is meant to reduce the number of Control Write MSGs exchanged using directed relay addressing.

The Get Mgr Domain Info Message packet exchange is meant to serve the following purposes:

- To exchange the state of the Manager Domains on both sides of a boundary link.
- To exchange the state of the boundary link at both ends
- To negotiate the owner of any boundary link Private IP Network based on the state of the Manager Domains and other data contained entirely within the message payload
- To enable the owner to negotiate the boundary link Private IP Network subnet number
- To enable the owner to allocate private network IP addresses to the non-owning manager
- To enable the validation and merger of the two Manager Domains

To accomplish these above tasks, the Get Mgr Domain Info messages have fields devoted to the following features:

Because of the atomic-like processes required to provide the above abilities in parallel all within one packet payload, the payload format has a mix of data about both managers in the same message. This statement applies to both the Get Mgr Domain Info Request and Get Mgr Domain Info Response message types, so care must be taken to keep straight which data are associated with which manager.

Therefore, for the discussion about the Get Mgr Domain Info Messages payload fields, the following definitions apply:

Sender: The field contains the value associated with the originator (the sender) of the Get Mgr Domain Info message, regardless whether it is a Get Mgr Domain Info Request or Get Mgr Domain Info Response.

Local: Same as *Sender*. The sending manager is the local manager.

Recipient: The field contains the value associated with the receiver (the recipient) of the Get Mgr Domain Info message, regardless whether it is a Get Mgr Domain Info Request or Get Mgr Domain Info Response.

Peer: Same as Recipient. The recipient manager is the peer manager.

Requestor: The field contains the value associated with the initiator (the sender) of the Get Mgr Domain Info Request Message which first defined the value. Such fields contained in a Get Mgr Domain Info Response Message shall contain the value reflected from that same field of the associated Get Mgr Domain Info Request.

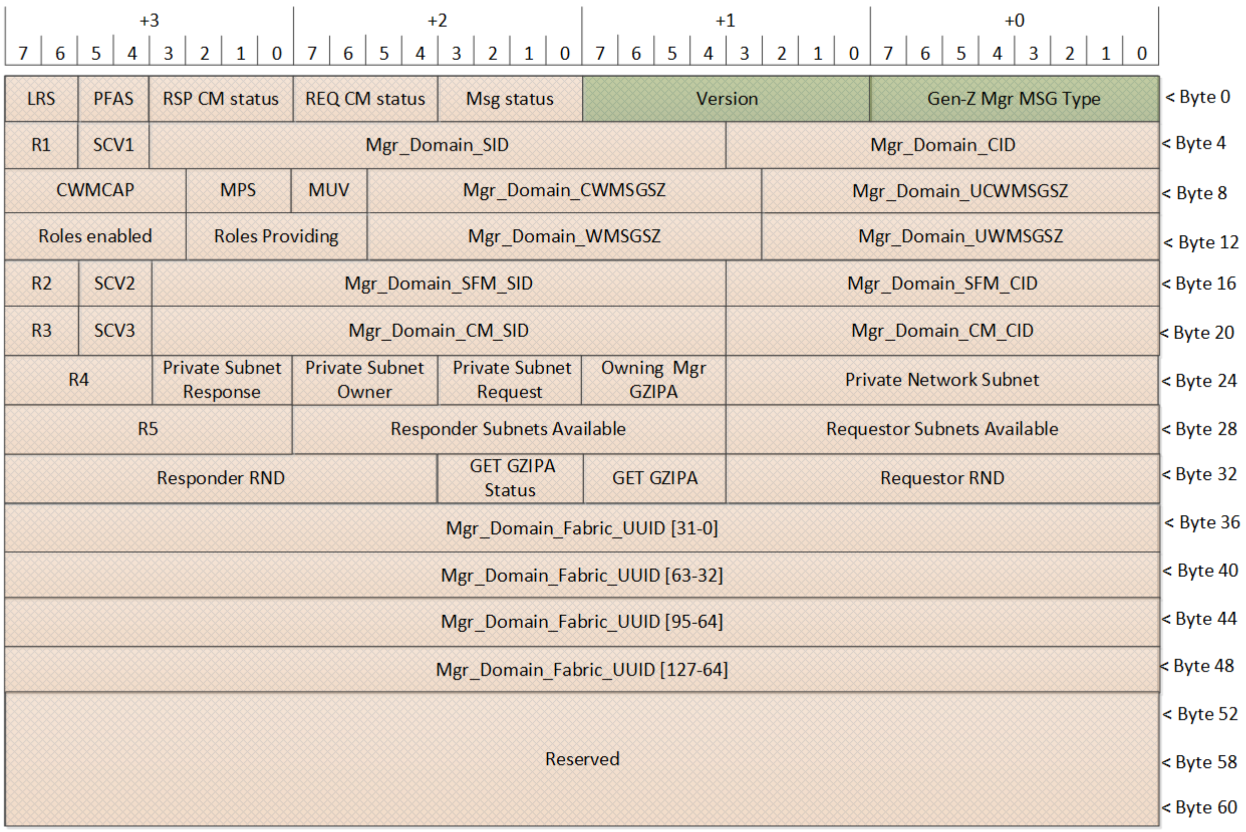
Responder: The field contains the value associated with the response payload of a prior Get Mgr Domain Info Response Message. Such fields contained in a Get Mgr Domain Info Request

Message shall contain the value reflected from that same field of the associated Get Mgr Domain Info Response.

Owner: The field contains the value selected by the manager chosen as the boundary link private network owner, as determined by the manager priority tie breaker rules specified in *Section* ***Error! Reference source not found. Error! Reference source not found.***. The Owner is fined by a Sender-relative Owner Status field. Each Sender will indicate itself or the peer manager as the 'owner'.

Sender and Recipient fields, if both present in a Get Mgr Domain Info Request and the associated Get Mgr Domain Info Response will swap values between the request and response messages. The 'names' are tied to the initiator of the specific packet.

Requestor and Responder fields, if both present in a Get Mgr Domain Info Request and the associated Get Mgr Domain Info Response will *not* swap values between the request and response messages. The 'names' are tied to the initiator of the Request/Response message exchange (the original requestor) so the values are appropriately named throughout the exchange.



+3 (7 6 5 4 3 2 1 0)			+2 (7 6 5 4 3 2 1 0)		+1 (7 6 5 4 3 2 1 0)	+0 (7 6 5 4 3 2 1 0)	
LRS	PFAS	RSP CM status	REQ CM status	Msg status	Version	Gen-Z Mgr MSG Type	< Byte 0
R1	SCV1	Mgr_Domain_SID				Mgr_Domain_CID	< Byte 4
CWMCAP	MPS	MUV	Mgr_Domain_CWMSGSZ			Mgr_Domain_UCWMSGSZ	< Byte 8
Roles enabled	Roles Providing		Mgr_Domain_WMSGSZ			Mgr_Domain_UWMSGSZ	< Byte 12
R2	SCV2	Mgr_Domain_SFM_SID				Mgr_Domain_SFM_CID	< Byte 16
R3	SCV3	Mgr_Domain_CM_SID				Mgr_Domain_CM_CID	< Byte 20
R4		Private Subnet Response	Private Subnet Owner	Private Subnet Request	Owning Mgr GZIPA	Private Network Subnet	< Byte 24
R5			Responder Subnets Available			Requestor Subnets Available	< Byte 28
Responder RND				GET GZIPA Status	GET GZIPA	Requestor RND	< Byte 32
Mgr_Domain_Fabric_UUID [31-0]							< Byte 36
Mgr_Domain_Fabric_UUID [63-32]							< Byte 40
Mgr_Domain_Fabric_UUID [95-64]							< Byte 44
Mgr_Domain_Fabric_UUID [127-64]							< Byte 48
Reserved							< Byte 52
							< Byte 58
							< Byte 60

Figure 5-6 Get Mgr Domain Info Message Payload Format

Table 5-4 Mgr Domain Info Message Payload Field Definitions

Field Name	Size & Type	Description
MSG Status	4 bits	If message is a Get Mgr Domain Info Request, the MSG Status field bits are encoded as follows: Bit 0: if 1b, some or all of requestor's Manager Domain values are included in payload of this request. Bit 1: if 1b, requestor's networking values are presented for negotiation or confirmation Bits 2-3: Reserved If message is a Get Mgr Domain Info Response, the 4-bit MSG Status field is encoded as follows: 0x0: Manager values of responding manager are included in payload. Responder's values were received with no errors. 0x1: Manager values of responding manager are included in payload. Responder's values were received with error. Retry requested. 0xC: Request Failed, invalid or inconsistent request 0xD: Request Failed, networking request error, invalid request, requestor not authorized 0xE: Request Failed, networking request error, invalid request, responder not authorized 0xF: Request Failed, unspecified reason (-1) Retry allowed. 0x2 – 0xE: Reserved Get Mgr Domain Info Response messages shall contain the responding manager's Manager Domain values and the manager's replies to all networking negotiation or confirmation requests included in the associated Get Mgr Domain Info Request message.
REQ CM Status	4 bits	Requestor CM Status: This field indicates the CM search status of the manager initiating the Get Mgr Domain Request message: 0x0: invalid status 0x1: Stand Alone domain, still configuring 0x2: Stand Alone domain, configuration complete

		0x3: CM identity unknown, CM expected 0x4: CM identity believed known, connection not yet made 0x5: CM identity known, connection made, this manager domain configuration not yet validated 0x6: CM known, has validated requestor manager domain configuration 0x7 – 0xf: Reserved The CM Status is used to set manager priority during manager to manager communications, per *Section 4.1 General Manager Domain Exploration Policies*
RSP CM Status	4 bits	Responder CM Status: This field indicates the CM search status of the manager initiating a Get Mgr Domain Response message as a reply to an associated request message. 0x0: invalid status 0x1: Stand Alone domain, still configuring 0x2: Stand Alone domain, configuration complete 0x3: CM identity unknown, CM expected 0x4: CM identity believed known, connection not yet made 0x5: CM identity known, connection made, this manager domain configuration not yet validated 0x6: CM known, has validated manager domain configuration 0x7 – 0xf: Reserved The CM Status is used to set manager priority during manager to manager communications, per *Section 4.1 General Manager Domain Exploration Policies*
PFAS	2 bits	Peer Fabric Acceptance Status: The Acceptance status of the recipients Manager Domain configuration as declared by the *sending manager's* CM that controls the *sending manager's* Fabric UUID is as follows: 0x0: Not yet validated, or no info from local CM 0x1: Validation pending by local CM 0x2: Validation attempted and failed by local CM 0x3: Validated and accepted by local CM

		If message is a Get Mgr Domain Info Request, this field is filled in by the Requesting Manager and indicates the Fabric Status of the responding manager's fabric according to the Requesting manager's CM. If a Response message, this field is filled in by the Responding Manager and indicates the Fabric Status of the Requesting manager's fabric according to the Responder's CM. A boundary link manager should not enable explicitly addressed packets to flow across a boundary link unless both boundary manager's domain PFAS values are set at 0x3 and both Fabric UUIDs are the same. See *Section* ***Error! Reference source not found. Error! ference source not found.***.
LRS	2 bits	Link Restriction Status: This field indicates the link restriction status at the sender's end of the boundary link: 0x0: Data Space and Control Space explicitly addressed packets allowed to ingress 0x1: Control Space explicitly addressed packets allowed to ingress 0x2: Only directed relay, unreliable Control Write MSG packets allowed to ingress 0x3: Reserved If message is a Get Mgr Domain Info Request, this field is filled in by the Requesting Manager and indicates the Link Restriction Status of the Requesting manager's port of the boundary link. If a Response message, this field is filled in by the Responding Manager and indicates the Link Restriction Status of the Responding manager's port of the boundary link. Managers may exchange the Mgr Domain Info Exchange messages at any time. The Link Restriction Status allows each manager to explicitly inform the other manager of the current state of the boundary link filters in place on their end of the link.
Mgr Domain CID	12 bits	Mgr Domain CID: the sending manager's CID

		If message is a Get Mgr Domain Info Request, and this field is valid, this is the Component ID of the Requesting manager. If a Response message, this is the Component ID of the Responding manager. The validity of this field is dependent upon the value of the SCV1 field.
Mgr Domain SID	16 bits	Mgr Domain SID: the sending manager's SID If message is a Get Mgr Domain Info Request, this is the Component Sub-net ID of the Requesting manager If a Response, this is the Component Sub-net ID of the Responding manager. The validity of this field is dependent upon the value of the SCV1 field.
SCV1	2 bits	SID CID Valid flags 1: these bits indicate whether the Mgr Domain CID and Mgr Domain SID fields are valid. 0x0: neither CID or SID fields are valid 0x1: Only CID field is valid 0x2: Reserved 0x3: both CID and SID fields are valid
R1	2 bits	Reserved
UCWMSGSZ	11 bits	UCWMSGSZ: sender's Unreliable, Control Write MSG Message Size If message is a Get Mgr Domain Info Request, and this field is valid, this is the max unreliable Control Write MSG size supported by the requesting manager. If a Response, and this field is valid, this is the max unreliable Control Write MSG size supported by the responding manager.
CWMSGSZ	11 bits	UCWMSGSZ: sender's reliable, Control Write MSG Message Size If message is a Get Mgr Domain Info Request, and this field is valid, this is the max reliable Control Write MSG size supported by the requesting manager. If a Response, and this field is valid, this is the max reliable Control Write MSG size supported by the responding manager. If CWMCAP[0] == 0b, this CWMSGSZ field is Reserved

MUV	2 bits	Mgr UUIDs Valid: This 2 bit field defines if the Sender's associated UUID fields are valid: Bit 0: Fabric-UUID Field is valid if this bit == 1b Bit 1: Reserved
MPS	3 bits	MPS: Sender's Maximum Payload Size of message packets. Encoded the same as Max Packet Payload in the Core CAP 1 Control register per Gen-Z Core Specification, Section 8.15.4. If message is a Get Mgr Domain Info Request with requestor Introduction Exchange information included (MSG Status[0] = '1b,) this field is the Max Packet Payload as enabled on the requesting manager. If MSG Status[0] == 0, this field is reserved. If a Response message, this field is the Max Packet Payload as enabled on the responding manager.
CWMCAP	5 bits	CWMCAP: Sender's Control Write MSG Capabilities. All in-band managers must support unreliable Control Write MSG and directed relay unreliable Control Write MSG. The LSB 3 bits of this field indicates which additional Control Write MSG and Write MSG capabilities are supported by the manager. These three LSBs are defined as follows: Bit 0: reliable Control Write MSG supported Bit 1: unreliable Write MSG supported Bit 2: reliable Write MSG supported The 2 MSBs of this field indicate whether the manager supports the optional Manager Message Types. Bit 3: Software Defined Payload Format supported Bit 4: HTTP Payload Format supported If message is a Get Mgr Domain Info Request with requestor information included, this field indicates the capabilities of the requesting manager. If this is a Response message, this field indicates the capabilities of the responding manager.

UWMSGSZ	11 bits	UWMSGSZ: Sender's unreliable, Write MSG Message size. If message is a Get Mgr Domain Info Request, and this field is valid, this is the max unreliable Write MSG size supported by the requesting manager. If a Response, and this field is valid, this is the max unreliable Write MSG size supported by the responding manager. The Write MSG packets are data space packets, but may be used by compatible managers once the two managers are integrated into a single Gen-Z CID Namespace and data space packets are enabled throughout. If CWMCAP[1] == 0b, this UWMSGSZ field is Reserved
WMSGSZ	11 bits	WMSGSZ: Sender's reliable Write MSG message size. If message is a Get Mgr Domain Info Request, and this field is valid, this is the max reliable Write MSG size supported by the requesting manager. If a Response, and this field is valid, this is the max reliable Write MSG size supported by the responding manager. The Write MSG packets are data space packets, but may be used by compatible managers once the two managers are integrated into a single Gen-Z CID Namespace and data space packets are enabled throughout. If CWMCAP[2] == 0b, this WMSGSZ field is Reserved
Roles Providing	5 bits	The Roles Providing flags indicate which Gen-Z management roles the sending manager is currently providing. The 5 bits are encoded as follows: Bit 0: Primary Manager and/or Primary Fabric Manager services provided Bit 1: Secondary Fabric Manager services provided Bit 2: Composing Services provided Bit 3: Reserved Bit 4: Reserved

		If a bit is set to 1b, the associated service or role is being provided to this manager domain by the sending manager. If bit 1 is set to 1b, the message recipient should look to the Mgr_Domain_SFM_CID and Mgr_Domain_SFM_SID fields. If these fields are equal to the Mgr_Domain_CID and Mgr_Domain_SID fields then this message comes from the Secondary Fabric Manager of the domain. In this situation, the GCID of the Primary Fabric Manager for this domain is not contained in this message.
Roles Enabled	5 bits	The Roles Enabled flags indicate which Gen-Z management roles the sending manager is currently enabled to provide. The 5 bits are encoded as follows: Bit 0: Primary Manager enabled Bit 1: Primary Fabric Manager enabled Bit 2: Secondary Fabric Manager enabled Bit 3: Composing Services enabled Bit 4: Reserved If a bit is set to 1b, the associated service or role is enabled. Bits 0 and 1 denote services that are available only to the manager domain of this manager. If bit 2 is set, the sending manager is capable of serving as a Secondary Fabric Manager to another domain. Composability Manager services may be provided to components and managers outside the sender's manager domain.
Mgr Domain SFM CID	12 bits	Secondary Fabric Manager CID of sender's domain. If message is a Get Mgr Domain Info Request, and this field is valid, this is the Component ID of the SFM installed in the Requesting manager's Domain. If a Response message, this is the Component ID of the SFM installed in the Responding manager's Domain.
Mgr Domain SFM SID	16 bits	Secondary Fabric Manager SID of sender's domain.

		If message is a Get Mgr Domain Info Request, and this field is valid, this is the Component Sub-net ID of the SFM installed in the Requesting manager's Domain. If a Response, this is the Component Sub-net ID of the SFM installed in the Responding manager's Domain.
SCV2	2 bits	SID CID Valid flags 1: these bits indicate whether the Mgr Domain SFM CID and Mgr Domain SFM SID fields are valid. 0x0: neither CID or SID fields are valid 0x1: Only CID field is valid 0x2: Reserved 0x3: both CID and SID fields are valid
R2	2 bits	Reserved
Mgr Domain CM CID	12 bits	Composability Manager CID of Sender's domain. If message is a Get Mgr Domain Info Request, and this field is valid, this is the Component ID of the Composability Manager known in the Requesting manager's Domain. If a Response, and this field is valid, this is the Component ID of the Composability Manager known in the Responding manager's Domain. A manager may have connections with a Composability Manager outside of the Gen-Z fabric, so the CM GCID may not exist.
Mgr Domain CM SID	16 bits	Composability Manager SID of Sender's domain. If message is a Get Mgr Domain Info Request, and this field is valid, this is the Component Sub-net ID of the Composability Manager known in the Requesting manager's Domain. If a Response, this is the Component Sub-net ID of the Composability Manager known in the Responding manager's Domain. A manager may have connections with a Composability Manager outside of the Gen-Z fabric, so the CM GCID may not exist.
SCV3	2 bits	SID CID Valid flags 1: these bits indicate whether the Mgr Domain CM CID and Mgr Domain CM SID fields are valid. 0x0: neither CID or SID fields are valid

		0x1: Only CID field is valid 0x2: Reserved 0x3: both CID and SID fields are valid
R3	2 bits	Reserved
Private Network subnet	12 bits	Private Network subnet: this is the 12 bit subnet number in the owning manager's private network it wishes to create or has created to enable communications across the boundary link carrying this current Request. This field value shall be between 3 and 4093. A value of 0x0 in this field indicates no valid subnet value is being proposed or is in use The status of this value is further defined by the Private Subnet Status and the AS fields
Owning MGR GZIPA	4 bits	Owning Manager Gen-Z IP Address: this is the 4 bit IP number of the owning manager's private network IP address. The Owning manager is the leader of the private network subnet negotiations, per *Section **Error! eference source not found. Error! Reference source not found.***. This field value shall be between 1 and 14. If this value is 0x0, the Manager has no valid GZIPA for this subnet or no owner has been established. The Private Subnet Owner field further defines the interpretation of the Owning MGR GZIPA field.
Private Subnet Request	4 bits	Private Subnet Request: this 4 bit field is the requesting manager's proposal for negotiations or confirmations that establish the Private Network Subnet number and the private network status. The Private Subnet Request values are defined as follows: 0x0: invalid subnet number, no request herein 0x1: subnet number is owner's proposal for subnet value If Private Subnet Owner != 0x1, this request code is Reserved. 0x2: subnet value has been accepted by non-owner This subnet value is pending for use

		0x3: subnet number is owner's proposal for subnet value Last subnet option, no other values remain If Private Subnet Owner != 0x1, this request code is Reserved. 0x4: subnet value which has been activated by requestor This subnet value is active or pending activation on the responder When both boundary managers have issued request messages with this Private Subnet Request code and received the appropriate, successful Private Subnet Response code in return, the boundary link Private Network is ready for transporting Control Write Messages with Ethernet formatted payloads. 0x5: subnet value owner requests responder to activate If Private Subnet Owner != 0x1, this request code is Reserved. 0x6: Reserved 0x7: subnet value which Owner is revoking. This subnet value will be reclaimed. If Private Subnet Owner != 0x1, this request code is Reserved. 0x8: subnet value which Non-owner is releasing. This subnet value will be reclaimed. If Private Subnet Owner != 0x2, this request code is Reserved. 0xE: Initiate Private Network Owner negotiation, but this requesting manager cannot or will not be the owner 0xF: Initiate Private Network Owner negotiation Sending manager has included new Requestor RND number and the Requestor Subnets Available field is valid 0x9 – 0xE: Reserved If Private Subnet Owner == 0x0, Private Subnet Request shall be = 0x0 or 0xF. No subnet number shall be

		negotiated before an owner of the subnet is determined. If Private Subnet Owner == 0x2, and this sender has accepted a prior subnet number proposal from the owner, this sender shall use one of { 0x02, 0x04, 0x08} request codes and indicate the associated subnet number in the Private Network Subnet field. If a Get Mgr Domain Info Response message, the responding manager shall reflect this field and its associated Private Network Subnet value for the Get Mgr Domain Info Request message in the response message.
Private Subnet Owner	4 bits	Private Subnet Owner: a 4 bit field set by the sending manager. This value is the sending manager's take on the owner of the boundary link Private Network: 0x0: No ownership resolution 0x1: Manager who sent this message is the owner 0x2: Manager who will receive this message is the owner 0x3: Owner conflict detected, no owner chosen, or no owner available. repeat ownership resolution process if appropriate This code shall be used only in a response message. 0x4 – 0xF: Reserved If this is a Request message with Private Subnet Request == (0xE or 0xF), the requestor shall set Private Subnet Owner == 0x0.
Private Subnet Response	4 bits	Private Subnet Response is a 4 bit response to the Private Subnet Request. If this is a Get Mgr Domain Request message, this field is Reserved If this is a Get Mgr Domain Response message, this field encodes the responding manager's reply as follows:

		0x0: no response. Only appropriate if Private Subnet Request == 0; 0x1: Accept owner's subnet value offer. Only appropriate if Private Subnet Request == 0x1 or 0x3 0x2: Accept owner's state confirmation / agree with non-owner's state declaration Only appropriate if Private Subnet Request == 0x2 or 0x4 0x3: Accept owner's state change request Only appropriate if Private Subnet Request == 0x5 or 0x7 0x4: Reject owner's offer Only appropriate if Private Subnet Request == 0x1 or 0x3 0x5: Reject owner's offer and request restart of process to select subnet number Only appropriate if Private Subnet Request == 0x1 or 0x3 0x6: Reject owner's state confirmation, owner to restart process to reconcile Only appropriate if Private Subnet Request == 0x2 or 0x4 0x7: Reject owners state change request, not ready Owner to retry Only appropriate if Private Subnet Request == 0x5 0x8: Accept owners request to revoke the given subnet value If and only if Private Subnet Request == 0x7, this response code is the only appropriate value to use. 0x9: Accept non-owner's request to release the given subnet value If and only if Private Subnet Request == 0x8, this response code is the only appropriate value to use. 0xa: Ownership mismatch: Responder does not agree with Requestor's value of Private Subnet

		Owner; Private Subnet Request rejected. Requestor to initiate new ownership resolution cycle. 0xE: Abdicate ownership to the requesting manager The responding manager cannot or will not act as the owner of the private network. This response packet shall contain the original Responder RND value and the appropriate Responder Subnets Available value. 0xF: Acknowledge request to initiate new ownership resolution cycle. This response packet shall contain the original Responder RND value and the appropriate Responder Subnets Available value. 0x9 – 0xD: Reserved
R4	4 bits	Reserved
Requestor Subnets Available	12 bits	Requestor Subnets Available value indicates the number of un-used subnets which the manager that issued this Get Mgr Domain Info Request has left to choose among. This count includes the current value of the Private Network Subnet field. This value shall be valid whenever the Private Network Subnet field value is not 0x0
Responder Subnets Available	12 bits	Responder Subnets Available value indicates the number of un-used subnets which the manager that issued this Manager Info Response message has left to choose among. This count includes the current value of the Requestor Network Subnet field if that value is available. This value shall be valid whenever the Requestor Network Subnet field value is not 0x0
R5	8 bits	Reserved
Requestor RND	12 bits	Requestor Random is a 12 bit field filled in with a random number generated by the Requestor of the message the first time a non-zero value is present in the Private Network Subnet field. The same random number shall be included in this field of all subsequent Get Mgr Domain Info Message requests issued by this manager whenever the Private Network Subnet field is non-zero.

		This value is used to break priority ties in the private subnet selection process, as described in *Section* ***Error! eference source not found. Error! Reference source not found.*** *.*
GET GZIPA	4 bits	GET GZIPA: This 4 bit field is contains the 4 LSBs of the Gen-Z IP Address being negotiated between the boundary link Private Subnet owner and non-owner. This field shall be non-zero only when Private Subnet Owner field == 0x1 or Private Subnet Owner == 0x2 This field is filled in by the sending manager. The interpretation of this value is dictated by the GET GZIPA Status field in this message.
GET GZIPA Status	4 bits	GET GZIPA Status: The sender of this message uses this 4 bit field to define how the GET GZIPA field of this message is to be interpreted. If this message is a Get Mgr Domain Info Request message, the Get GZIPA Status field is decoded as: 0x0: Get GZIPA field invalid, no GZIPA is being requested, no changes being proposed 0x1: If Private Subnet Owner == 0x2, Get GZIPA field invalid, new GZIPA is requested If Private Subnet Owner != 0x2, this status value is Reserved (only the non-owner can make a request for a new GZIPA) 0x2: If Private Subnet Owner == 0x2, Get GZIPA field valid, sender of message is requesting this GZIPA be granted to it If Private Subnet Owner != 0x2, this status value is Reserved (only the non-owner can make a request) 0x3: If Private Subnet Owner == 0x2, Get GZIPA field valid, contains the currently in use Gen-Z IP Address of the sending manager. No new GZIPA request is being made. If Private Subnet Owner != 0x2, this status value is Reserved. (only a non-owner *confirms* its Gen-Z IPA in a Request message.) 0x4: If Private Subnet Owner field == 0x1, the Get GZIPA field is valid, the owner is revoking this Gen-Z IP address from the non-owner recipient.

		If Private Subnet Owner field != 0x1, this status value is Reserved. 0x5: If Private Subnet Owner field == 0x2, the Get GZIPA field is valid, and the non-owner is releasing this Gen-Z IP address. If Private Subnet Owner field != 0x2, this status value is Reserved 0x7-0xF: Reserved If this message is a Get Mgr Domain Info Response message, the Get GZIPA Status field is decoded as: 0x0: Get GZIPA field invalid, no GZIPA is being granted, no GZIPA in use 0x1: If Private Subnet Owner Field == 0x1, Get GZIPA field is invalid, no GZIPA available or requested GZIPA is un-available If Private Subnet Owner Field != 0x1, this status value is Reserved. Only a valid response for requests == 0x1 and 0x2 0x2: If Private Subnet Owner Field == 0x1, Get GZIPA field is valid, this value is being offered to Request originator by the owner If Private Subnet Owner Field != 0x1, this status value is Reserved. Only a valid response for requests == 0x1 and 0x2 0x3: If Private Subnet Owner Field == 0x1, Get GZIPA field is valid, this value is confirmed as the one currently assigned to the recipient (non-owner) manager. The sender is the network owner. If Private Subnet Owner Field != 0x1, this status value is Reserved. Only a valid response for request == 0x3 0x4: If Private Subnet Owner Field == 0x2, Get GZIPA field is valid, the non-owner manager is surrendering this value. If Private Subnet Owner field == 0x1, this status value is Reserved. Only a valid response for request == 0x4

		0x5: If Private Subnet Owner Field == 0x1, Get GZIPA field is valid, the owner accepting the surrender of the IP Address from the non-owner 0x6: If Private Subnet Owner == 0x1, the Get GZIPA field is valid and is the one the requestor wants to confirm. Owner does not confirm. Recipient (the non-owner) shall void its use of the given Gen-Z IP Address and make a new request. If Private Subnet Owner ==0x1, this status value is Reserved Only a valid response for request == 0x4 0x7-0xF: Reserved
Responder RND	12 bits	Responder Random is a 12 bit field filled in with a random number generated by the responder of the message any time a non-zero value is present in the Private Network Subnet field. This value is used to break priority ties in the private subnet selection process, as described *Section* ***Error! eference source not found. Error! Reference source not found.*** *.*
Mgr Domain Fabric _UUID	128 bits	Manager Domain Fabric UUID is the 128 bit UUID assigned by the CM (keeper of the GCID Namespace) as the UUID for the Gen-Z fabric which contains the sending manager's Manager Domain. Managers connected via Gen-Z fabric that have the same Fabric UUID are controlling components that are members of the same GCID namespace. Software and the Fabric Architect are responsible for preventing Manager Domains from using different GCID namespaces and having duplicate Fabric UUIDs.

Acknowledgments

The following individuals contributed to the development of the Gen-Z Management Architecture Specification:

Russ Herrell	Tom Vaden	Lisa Pallotti	Jeff Hilland
Jim Hull	Nilakantan Mahadevan	Barbara Craig	Betty Dall
John Mayfield	Duane Voth	Tim Symons	Tracy Spitler
Keith Shaw	Shyamkumar Iyer	Dimitri Sivanich	Brad Burke
Sudhir Shetty	Geoff Schunicht	Jerry Harrow	Hiren Patel
John Bockhaus	Derek Sherlock	Bill Scherer	Michael Krause
Kurt Schwemmer	Michael Ruan	Jeff Plank	Edlic Yiu
Michael Gregoire	Richard Brunner	Erich Hanke	Tina Rogers
Paul Lodrige	Kevin Marks	Kurtis Bowman	