In this document, we describe different use cases for Compute Express Link (CXL) 2.0 Fabric Manager and the mechanism for integrating CXL Fabric Manager with the [Open Fabric Management Framework](https://www.openfabrics.org/openfabrics-management-framework/) (OFMF) services developed by Open Fabrics Alliance. Readers unfamiliar with CXL may review the introduction section below, otherwise you may skip to CXL Fabric Manager use cases section 2. Details about OFMF services can be found [here](https://www.openfabrics.org/openfabrics-management-framework/).

1. Introduction

In this section, we briefly introduce CXL. CXL is a high-performance interconnect technology specification for connecting peripheral devices that can be either traditional non-coherent IO devices, memory devices or accelerators. It leverages the PCIe architecture to support coherent memory access capabilities between a host CPU and accelerators to meet the requirements of emerging data-intensive applications such as artificial intelligence, machine learning and graph analytics which benefit from accelerators. The cache-coherent interconnect allows resource sharing for higher performance, reduced software stack complexity, and lower system cost.

CXL defines three protocols that can be multiplexed together:

* CXL.io: enhanced version of PCIe 5.0 protocol for device discovery, initialization, link-up, and enumeration. It provides non-coherent load-store interface for IO devices.
* CXL.cache: defines interactions between a host and a device, allowing attached CXL devices to efficiently cache host memory
* CXL.mem: provides host processor with access to memory of attached CXL devices

Based on the multiplexed protocols, we can define three different types of devices attached to a CXL host: (1) Type 1 device consists of an accelerator with fully coherent cache implemented using CXL.io and CXL.cache protocols, (2) Type 2 device consists of an accelerator with its own memory (e.g., DDR, HBM) in addition to coherent cache, implementing CXL.io, CXL.cache and CXL.mem protocols, (3) Type 3 device is a memory expander for the host and consists of memory device attached to host via CXL.io and CXL.mem protocols.

1. CXL Fabric Manager and use cases

CXL Fabric Manager is responsible for querying and configuring the hardware resources within the fabric. It is a logical process which can run either on a host machine, BMC or CXL device/switch. It provides the functionality to bind hosts to devices and reconfigure switches to create composed servers. We envision multiple Fabric Manager entities running concurrently and managing their own pool of devices and ports. The state from across the different Fabric Managers needs to be aggregated and configured by the northbound services using a common interface in the software stack. The infrastructure provided by OFMF services provides such a common interface. Note that other fabrics and their fabric managers (e.g., Ethernet, InfiniBand) can also be concurrently running alongside the CXL Fabric Manager instances. OFMF provides the common interface to query and configure fabrics managed by *all the underlying fabric managers*.

CXL 2.0 adds support for a CXL switch, which enables us to drive fan-out of memory devices for memory scaling and expansion. Based on recommendations from the CXL consortium, we focus on the following target use cases for fabric-attached memory (FAM) pooling in CXL 2.0: (1) Single logical device where a FAM node’s resources consist of a single memory region bound to a host, (2) Multi logical device (MLD) where FAM node resources may be partitioned into multiple memory regions, each of which is bound to a different host.



1. Workflows for CXL 2.0 use cases

Below, we discuss the management operations carried out by Fabric Manager (FM) to enable the above use cases.

* **Initial resource discovery workflow**



|  |  |
| --- | --- |
| Use Case Description | Initial discovery of CXL switch and attached resources by CXL Fabric Manager |
| Actors | CXL Fabric Manager (FM), CXL Provider and OFMF service |
| Description | * FM discovers the connected switch and FAM node, and then initiates management commands to discover capabilities of the attached devices
* FM propagates the discovered resource information to Provider and OFMF service.
* After discovery, FM is ready for event notifications (e.g., if there is a hot removal of FAM node) and can enable host to FAM node binding.
 |
| Comments | Once the system components including switch and FAM node are powered up, they can be discovered (irrespective of the host power status). BMC and the host have a management link using MCTP or Redfish host interface. Here, we focus on the discovery of capabilities of attached FAM node and propagation of this information to upstream provider and OFMF service. At the end of this workflow, FAM node is ready for binding to a host. |
| Input Data | MCTP endpoint information for connected physical resources including the CXL switch and FAM node. FM initiates the discovery workflow for each endpoint.  |
| Preconditions | * BMC, switch and FAM node are in-service with active management network links (e.g., on SMBus, PCIe VDM at the physical layer)
* CXL switch is released from reset and loads its initial configuration from non-volatile memory. Ports are released from reset to link up.
* FM is up and communicating with devices using MCTP. Switch and FAM nodes implement FM APIs.
* FM has sent event notification through its Provider to the OFMF Provider Manager, so that OFMF is actively managing the FM.
 |
| Postconditions | * Provider data store contains the description of CXL switch, FAM node and associated properties (e.g., number of ports on switch, memory capacity on FAM node).
* OFMF Redfish tree contains the newly discovered resource information.
* State of the fabric matches the state stored in Provider and OFMF service data stores
 |
| Trigger | Once the switch and FAM node are connected over the management link, device enumeration on the physical bus is performed by the MCTP discovery protocol. Completion of MCTP discovery triggers the FM discovery workflow |
| Normal Flow  | FM detects the new resources and their property information, and this is percolated to OFMF services for querying by OFMF clients* Initial MCTP discovery protocol detects switch and FAM node endpoints
* FM issues *Get Supported Logs* command to get device specific logs (identified by UUID) on the switch and FAM node.
* For each log ID, FM issues *Get Command Effect Log* command which outputs the supported command opcode along with description of the command effects for the opcode.
* A switch is detected, so FM issues *Identify Switch Device* command to get a response of switch capabilities and capacity configuration, including physical port count, port ID of the management port on the switch, number of virtual CXL switches, etc.
* FM queries individual port information on the switch by issuing *Get Physical Port State* with the number of ports requested and port ID list. Response includes port specific information including whether it’s an upstream port (connected to host) or downstream port (connected to devices), along with connected device CXL version, type and number of supported logical devices.
* For each host port, FM issues *Get Virtual CXL Switch Info* with the number of virtual CXL switches (VCS) for which information is requested, along with their IDs. Response includes the count of VCS returned, whether each VCS is enabled/disabled, host binding status, etc.
* Since a FAM node is detected, FM checks if it is a MLD by issuing *MLD Component Command Set*, to get number of supported logical devices, per-logical device information including memory size, allocation region, etc.
* FM finally sets the event notification policy to receive events. *Get Event Interrupt Policy* checks the current event notification policy and is modified using *Set Event Interrupt Policy*, if needed.
* FM updates the newly discovered switch and FAM node in its data store. The data store uses native fabric data model based on Redfish CXL extensions.
* CXL provider queries the FM data store for new fabric state information in its next polling cycle. The newly discovered switch and FAM node are returned by FM to CXL provider, in response to the provider query.
* CXL provider translates the representation of the newly received fabric state, from its native data model based on Redfish CXL extensions to standard Redfish model (with no CXL-specific entities).
* CXL provider sends event notification to OFMF Resource Inventory service about the updated fabric state.
* OFMF Resource Inventory service updates its data store with the newly discovered resource information. Clients can now query OFMF for these resources.
 |

* **[WIP] Binding host to LD on a switch**



|  |  |
| --- | --- |
| Use Case Description | FM configures CXL switch to bind a host to memory region in FAM node  |
| Actors | FM, CXL 2.0 switch, FAM node, host |
| Description | * After initial resource discovery, CXL FM receives a request to bind a host connected to the switch, with memory region in FAM node attached on the same switch.
* FM configures the virtual CXL switch ports to the downstream physical ports, for binding to the FAM node
* If MLD device, CXL FM configures the logical device binding
 |
| Comments |  |
| Input Data | FM receives request from CXL provider for the host and LD to bind. Input includes: * Physical port where the device is attached
* Virtual CXL switch (VCS) ID, mapped 1:1 to each host
* Virtual PPB index within VCS
* In case of MLD, LD ID for the memory chunk
 |
| Preconditions | * BMC, switch, host and FAM node are in-service with active management network links (e.g., SMBus, PCIe)
* Initial resource discovery is complete
* The host and FAM memory region are unbound (if binding already exists, OFMF service can handle the client request)
 |
| Postconditions | * CXL switch configured with binding for host to logical device on FAM node
* Binding information percolated to the OFMF service layer
 |
| Trigger | Binding request from CXL provider |
| Normal flow | TBD |

* **[WIP] Unbinding host and LD on a switch** 
* **[WIP] Hot add of device**



* **[WIP] Managed hot removal from an unbound port**

