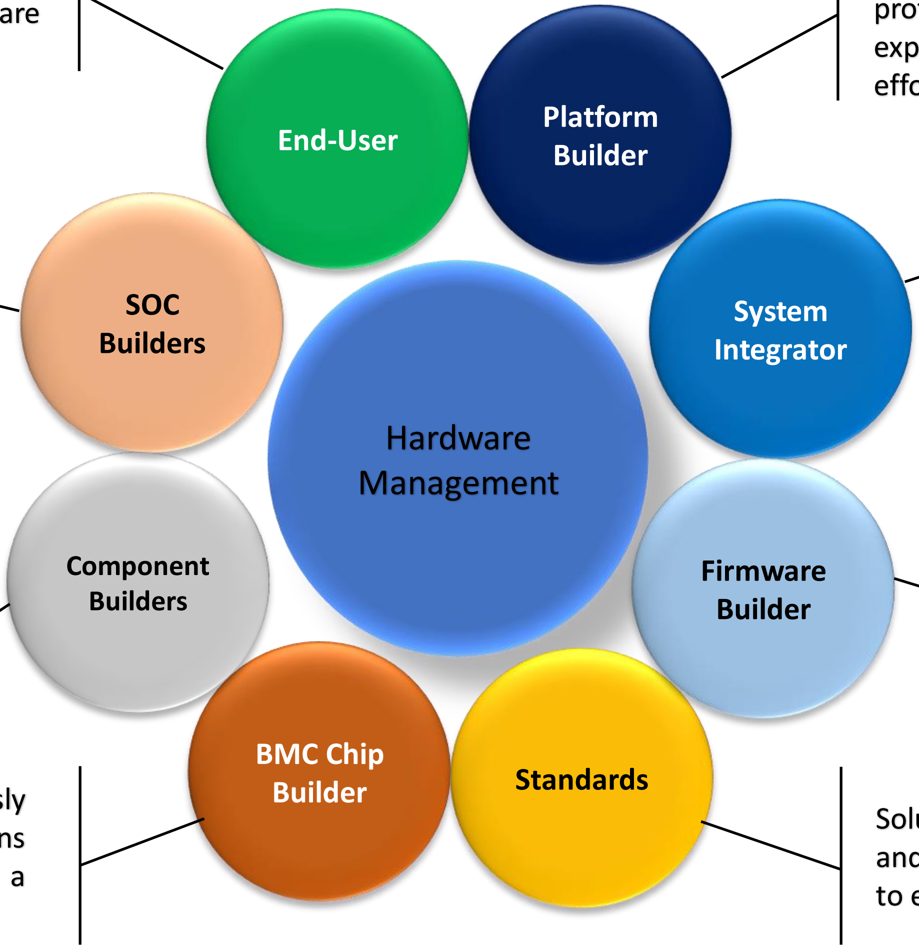
Project Phoenix

Open bmc firmware development kit suite

haveuthoughtaboutit@gmail.com

Introduction

Stakeholders of Hardware Management



End-Users input is critical to set their clear expectations, directions and requirements for the Hardware Management Solution.

SOC builders need for adding System Management interfaces through a standard communication path to the on-board Management Controller (BMC)

Intelligent-managed component builders need to easily develop the Firmware Drivers to interface with On-board Management Controllers

BMC Chip Builders need for seamlessly adding the interfaces and functions that are necessary for launching a Management solution.

Platform Builders ability to input their platform definition, thermal and power profile including provisioning expectations with minimum learning and effort.

System Integrators ability to integrate their choices with the Server/Hardware Management solution

Firmware builders can utilize the standard framework to develop extensions and product-specific features rather than build-from-scratch approach.

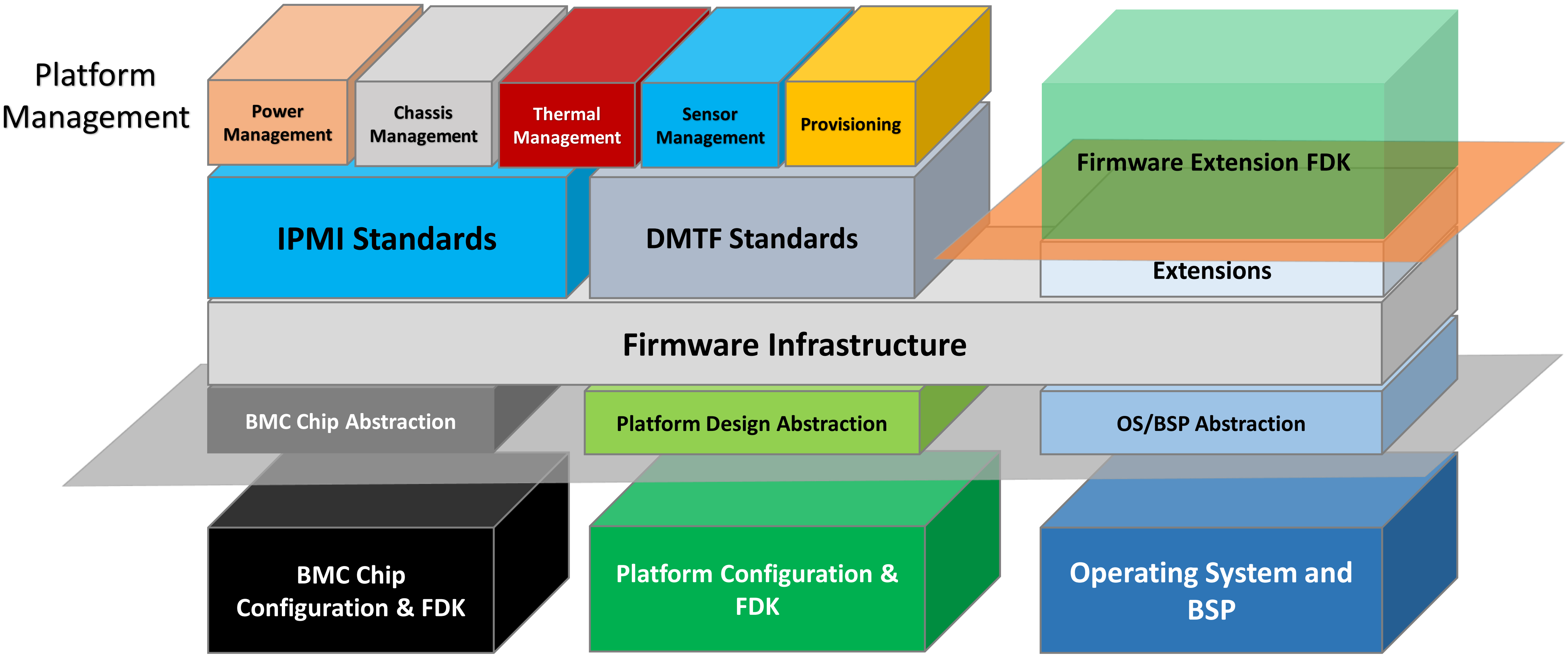
Solution and Orchestration alignment and compliance, along with robustness to evolving standards

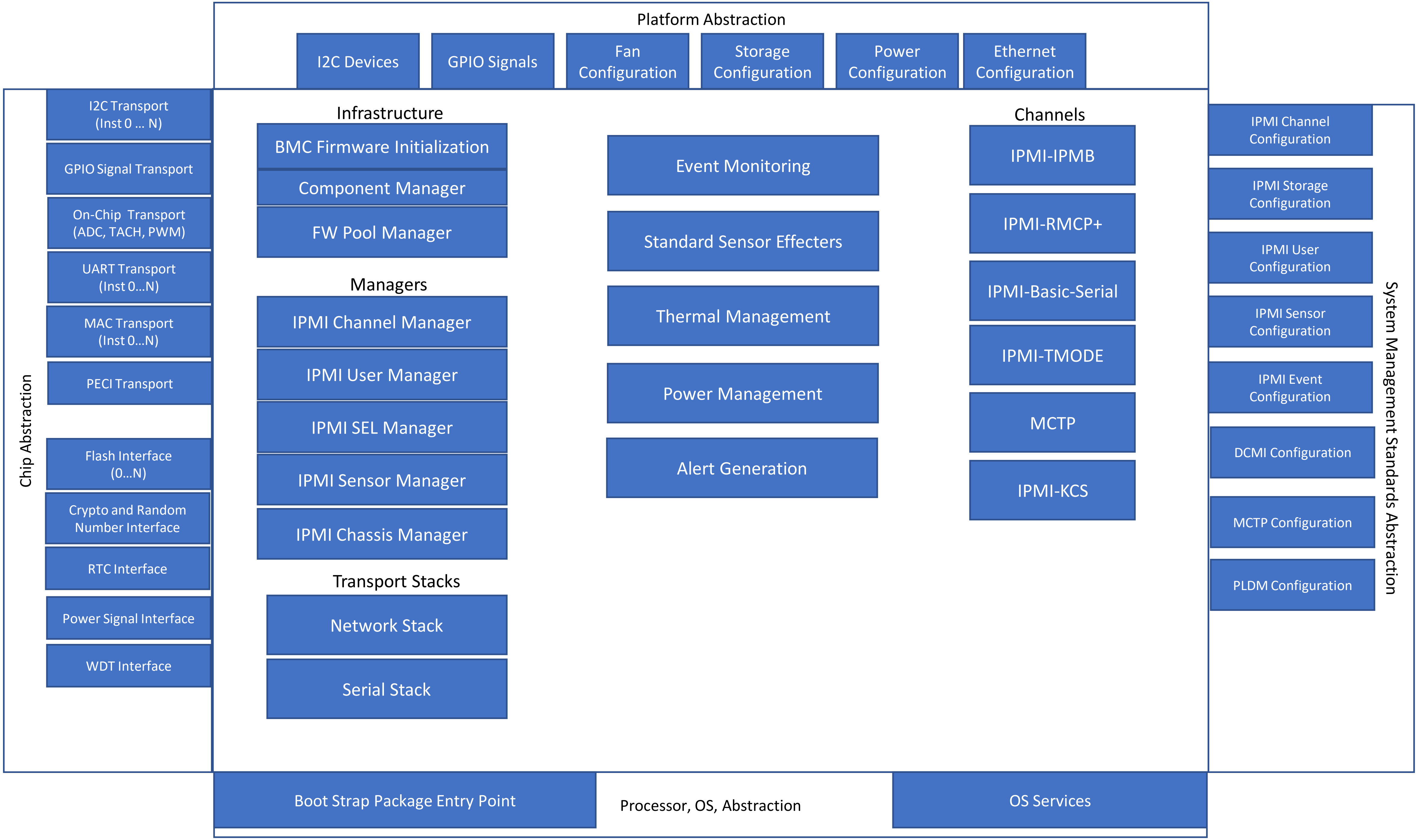
Architecture

overview

Features

Open BMC Firmware Building Blocks





Platform Abstraction

I2C Devices

GPIO Signals

Fan Configuration

Storage Configuration

Power Configuration

Ethernet Configuration

Chip Abstraction

I2C Transport (Inst 0 … N)

GPIO Signal Transport

On-Chip Transport (ADC, TACH, PWM)

UART Transport (Inst 0…N)

MAC Transport (Inst 0…N)

PECI Transport

Flash Interface (0…N)

Crypto and Random Number Interface

RTC Interface

Power Signal Interface

WDT Interface

Infrastructure

BMC Firmware Initialization

Component Manager

FW Pool Manager

Managers

IPMI Channel Manager

IPMI User Manager

IPMI SEL Manager

IPMI Sensor Manager

IPMI Chassis Manager

Transport Stacks

Network Stack

Serial Stack

Event Monitoring

Standard Sensor Effecters

Thermal Management

Power Management

Alert Generation

Channels

IPMI-IPMB

IPMI-RMCP+

IPMI-Basic-Serial

IPMI-TMODE

MCTP

IPMI-KCS

System Management Standards Abstraction

IPMI Channel Configuration

IPMI Storage Configuration

IPMI User Configuration

IPMI Sensor Configuration

IPMI Event Configuration

DCMI Configuration

MCTP Configuration

PLDM Configuration

Boot Strap Package Entry Point

Processor, OS, Abstraction

OS Services

Overview

- The solution provides the ability for customers who have the capability to develop their own commercial solutions for Server/Hardware Management and need development kits that would help them to move directly to their product specific requirements rather than spend time on standard components and features.
- The Firmware Development Kit (FDK) suite includes

❑S4H Open BMC Firmware Engine

❑S4H Open BMC Chip Configuration Firmware Development Kit (FDK)

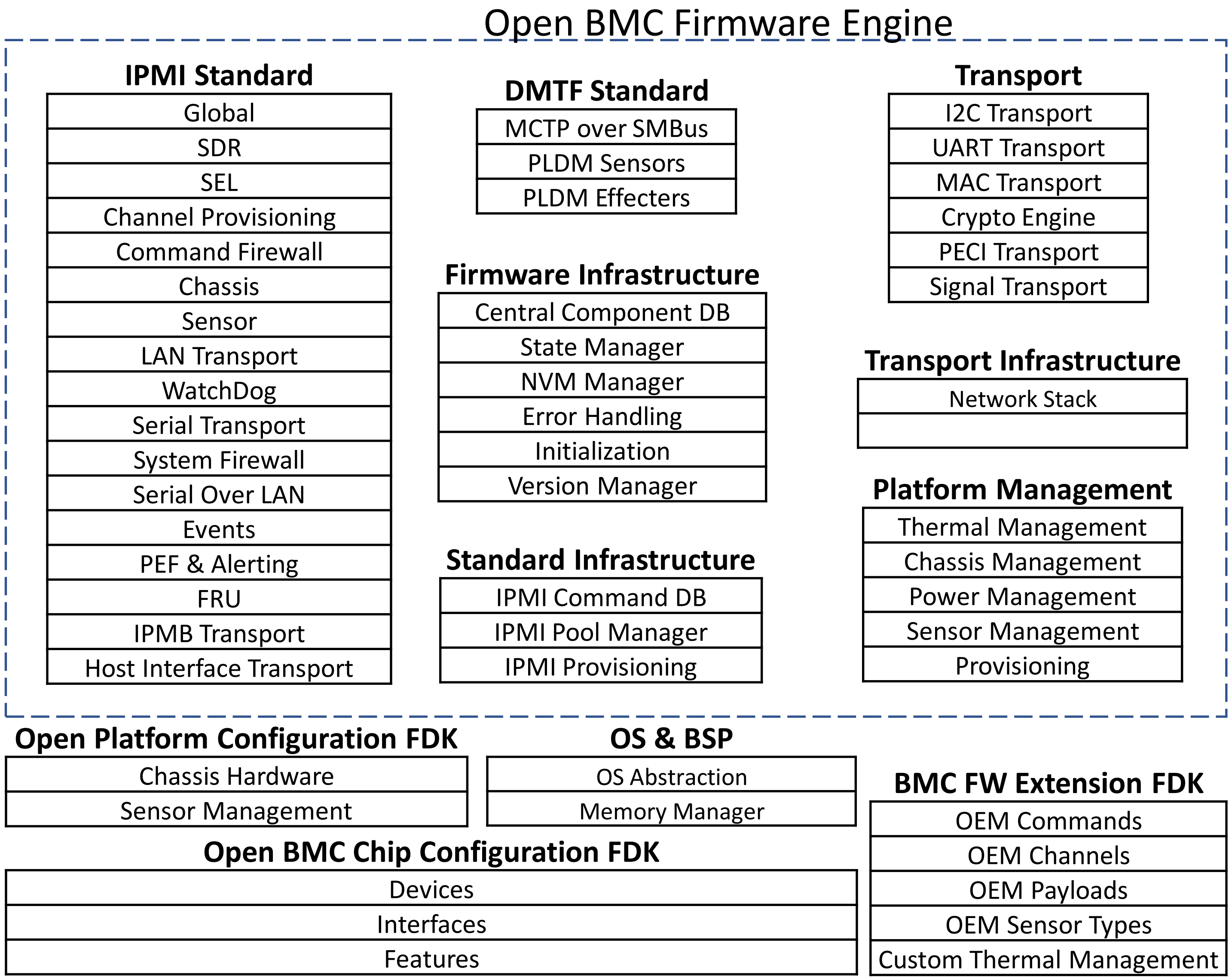
❑S4H Open Platform Management Configuration Firmware Development Kit (FDK)

❑S4H Open BMC Firmware Extensions Firmware Development Kit (FDK)

❑S4H Open BMC Test Framework Support

- The firmware development kits are created with the view to easily adopt to any BMC Chips and Platform configurations and also allow any product-specific extensions to be done with minimal effort.

Open BMC Firmware FDK Suite View



Open BMC Firmware Engine

IPMI Standard

- Global
- SDR
- SEL
- Channel Provisioning
- Command Firewall
- Chassis
- Sensor
- LAN Transport
- WatchDog
- Serial Transport
- System Firewall
- Serial Over LAN
- Events
- PEF & Alerting
- FRU
- IPMB Transport
- Host Interface Transport

DMTF Standard

- MCTP over SMBus
- PLDM Sensors
- PLDM Effecters

Firmware Infrastructure

- Central Component DB
- State Manager
- NVM Manager
- Error Handling
- Initialization
- Version Manager

Standard Infrastructure

- IPMI Command DB
- IPMI Pool Manager
- IPMI Provisioning

Transport

- I2C Transport
- UART Transport
- MAC Transport
- Crypto Engine
- PECI Transport
- Signal Transport

Transport Infrastructure

- Network Stack

Platform Management

- Thermal Management
- Chassis Management
- Power Management
- Sensor Management
- Provisioning

Open Platform Configuration FDK

- Chassis Hardware
- Sensor Management

OS & BSP

- OS Abstraction
- Memory Manager

Open BMC Chip Configuration FDK

- Devices
- Interfaces
- Features

BMC FW Extension FDK

- OEM Commands
- OEM Channels
- OEM Payloads
- OEM Sensor Types
- Custom Thermal Management

OPEN BMC Firmware Engine

OS & Boot Support Package Abstraction

Firmware infrastructure

Orchestration standards & infrastructure

Transport infrastructure

Platform Management

BMC Chip Configuration Abstraction

Platform Configuration Abstraction

Firmware Extensions Abstraction

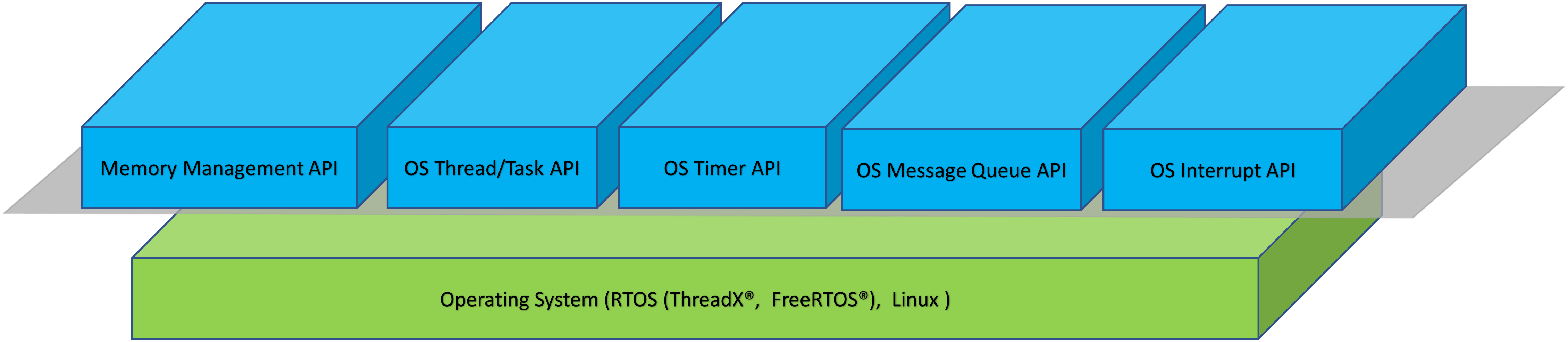
Firmware Engine Goals

- Firmware components shall utilize ZERO Copy Policy
- Firmware components shall be discoverable and establish a clear dependency to ensure all error propagation are limited
- All firmware components shall use a Non-blocking approach of utilizing message queues and no semaphores.
- All error validation is performed at the ingress point

OPEN BMC Firmware Engine

OS & Boot Support Package Abstraction

OS & Boot Support



❑Single Entry Point for Operating System/Bootstrap Package to invoke the BMC firmware Engine

OPEN BMC Firmware Engine

Firmware Infrastructure

Overview

- Central Component Database
  - The database which is used to discover, connect, and communicate between all the firmware, platform and BMC Firmware components
- State Management
  - The runtime component that is responsible to keep the states of all the runtime components with respect to the initialization, power transitions and errors
- NVM Management
  - The runtime component that is responsible for providing persistent storage for runtime components
- Error Handling
  - The runtime component that is responsible for handling all the information, warnings and errors that comes of each of the runtime components
- Version Management
  - The runtime component that is responsible for preserving and matching the versions of all the components and also responsible compatibility

Central Component Database

Each of the Run-time Component is assigned a unique 32-bit ID represents

- Entity Type : 31-24 bit : Firmware, Platform, BMC Chip
- Component Type : 23-16 bit : Sub type ID
- Function Type : 15-08 bit : Function Type
- Instance : 07-00 bit : Instance

APIs

- Component Registration APIs
- Service Thread/Task APIs
- Service Queue APIs
- Service Timer APIs
- Service NVRAM APIs
- State Info APIs
- Error APIs
- Warning APIs
- Information APIs

Firmware Component DB Entry

Next Firmware Component Link
Next Power State Component Link
Next Dependent Component Link
Next Always On Component Link
Component ID
Component State
Component Major Version
Component Minor Version
Service Thread/Task Information
Service Queue Information
Service Timer Information
Service NVRAM
Run-time State Information
Component Error Buffer
Component Warning Buffer
Component Information Buffer

Platform Component DB Entry

Next Platform Component Link
Next Platform Device Link
Component ID
Configuration Data Information
BMC Chip Transport Data List

BMC Chip Component DB Entry

Next BMC Chip Component Link
Next BMC Chip Interface Link
Component ID
BMC Chip Configuration Data

Central Component Database – CONT…

- All Thread/Tasks communicate using Service Message Queues
- Each Thread/Task shall update the State Manger information to show Heartbeat
- The Service Message Queues is a 32-bit data that carries different messages using at the MSB
  - Interrupt Message Types - Sent from the Interrupt Context to a Runtime Task/Thread
  - Timer Message Types - Send from the Timer Context to a Runtime Task/Thread
  - State Management Types - Send from the State Management
  - Power Transition Types - Send from Power Transitions
  - ASYNC Notification Message Types - 24-bits used for acknowledging different ASYNC Notifications
  - Service Message Types - 24-bit Message Reference to send Service Command and Response
  - Configuration Message Types - 24-bit Message Reference to send Configuration Data
  - Upstream Message Type - 24-bit Frame Reference to send upstream
  - Downstream Message Type - 24-bit Frame Reference to send downstream

State Management

- The State Management is a Infrastructure Run-time Thread/Task that is responsible for maintaining the Central Component Database at runtime.
- Run Time State Management
  - RTSTATE_PRE_INIT – Pre Initialization includes Registration of the Firmware, Platform and BMC Chip Components
  - RTSTATE_INIT – Initialization Process for each Component
  - RTSTATE_STARTED – Component Started
  - RTSTATE_STOPPED – Component Stopped due to Error
  - RTSTATE_SUSPENDED – Component Suspended due to Power State
  - RTSTATE_BLOCKED – Component Blocked for a ASYNC operation to complete
- Component State Data
  - As a way to capture the state of the Component, all Runtime state data shall be stored in a State Data Buffer registered with the Central Component Database
  - The Component State Data helps to capture the state of the Firmware when a critical error is encountered similar to Crash dump.

NVM Management

- The Non-volatile Storage is service provided for each component which registers a local RAM buffer and the size of the Non-volatile Storage requirement using the Component ID.
- The NVM Management provides a run-time Task/Thread Service to keep the local RAM buffer copy in sync with the Non-volatile Storage data
- The NVM Management does a lazy write to the Non-Volatile Storage
- NVM Management depends on the Platform FDK Non-Volatile Storage definition that includes the BMC Chip Flash definition that is utilized for completion of the read and write transactions

- NVM Management provides a set of APIs that will be used for registration, read and write operations

Error Handling

- The error handling is considered as essential service for each of the components registered with the Central Component Database
- The error handling can be utilized to track errors, warnings and information within the scope of the component
- The size of the buffer for errors, warnings and information is controlled by the Central Component Database

- Error Handling APIs are provided for recording CRITCAL_ERROR, ERRORs, WARNINGs and INFORMATION. All CRITCAL_ERROR will stop the task/thread and the State Management will propagate this error to all dependent components

Version Management

- Version Management is an essential primitive that helps to set clear run-time guidance on the compatibility between different components.
- Major and Minor versions are preserved at the Component level and checked when the registration is done

OPEN BMC Firmware Engine

Orchestration Standards and Infrastructure

Orchestration Standards

IPMI

❑Almost all the APIs of Intelligent Platform Management Interface (IPMI) 2.0 specification is available with options to configure the optional commands and the command parameters

❑IPMB, IPMI-KCS, IPMI-RMCP+/RMCP, IPMI over Serial, IPMI-TMODE are supported IPMI Communication interfaces

DCMI 1.5

Almost all the APIs additional described in the Data Center Management Interface (DCMI) 1.5 specification is available

DMTF MCTP, PLDM

❑DMTF Management Control Transport Protocol (MCTP) is supported over SMBus for all internal satellite controllers such as SOC

❑DMTF Platform Level Data Model (PLDM) is supported for MCTP for Sensors, Effecters communication exchange with satellite controllers such as SOC

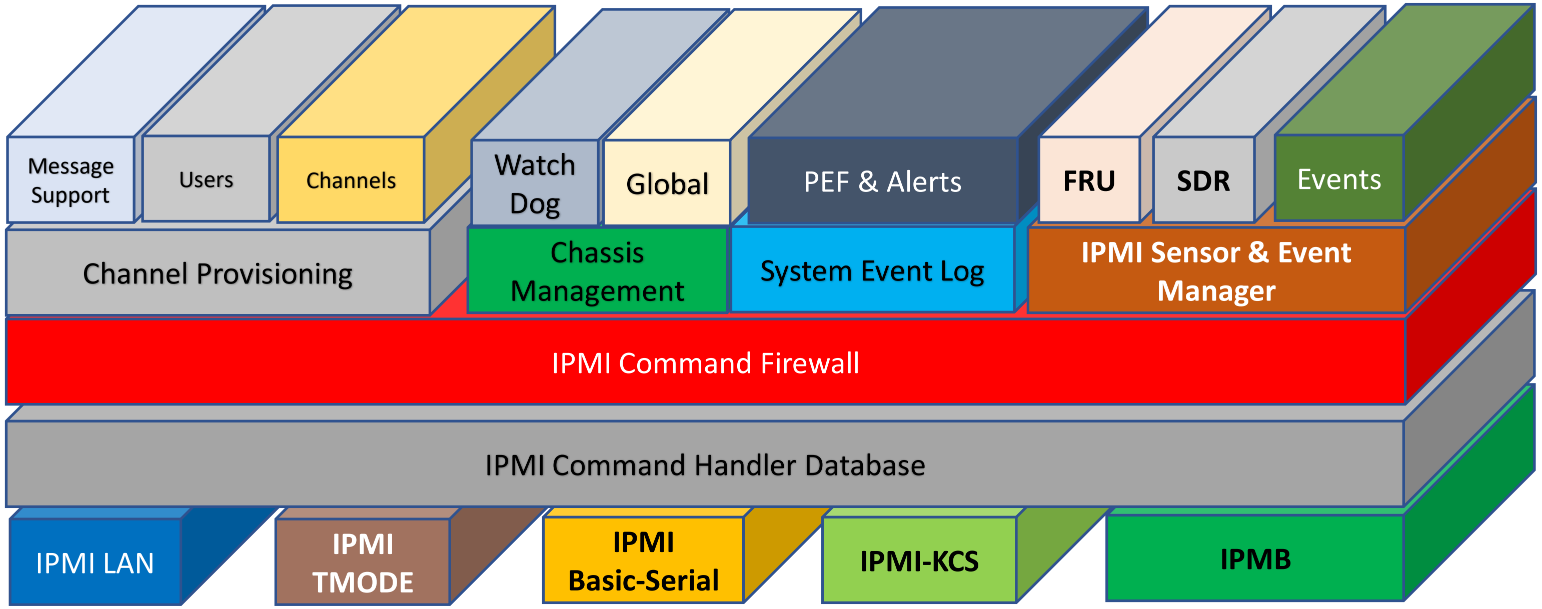
IPMI to PLDM Mapping

IPMI to PLDM Mapping of Sensors are done seamlessly and the DMTF MCTP is considered as a named OEM Channel

Overview – IPMI Standards

- IPMI Command Database: The database provides a provision to register the Command Handlers, along with the required transport privilege levels and the accepted IPMI Transports from the command originates.
  - All query commands are directly handled by IPMI Communication transport layers and the selective few that deals with Channels, Users and provisioning different IPMI layers are dealt through IPMI Provisioning Task, a single run-time task/thread
- IPMI Provisioning Task: The task that provides ability to streamline IPMI Command processing that is common to all the transports
- IPMI Communication Transports: The communication transports are defined through the Platform Configuration definitions and currently supports

IPMI Command Processing



- ❑Fully compliant with IPMI 2.0 and DCMI 1.5 with support for almost all commands of IPMI 2.0 and DCMI 1.5 Specifications (Exceptions will be listed in Release documents)
- ❑Support for IPMI Channels such as IPMB, LAN (RMCP, RMCP+), Serial (Basic-Serial and TMODE) and System Interface (KCS)
- ❑Support for IPMI and DCMI features such as Serial Over LAN, System Event Log, Sensor Data Repository (SDR), Field Replacement Unit (FRU), Chassis Management, Platform Event Filtering, Events and Command Firewall

IPMI Command Database

- Supports up to 256 commands and command interfaces
- Each Command Interface attributes are
  - Net Function , Network Command
  - CMD Extension (Utilized for DCMI and other extensions)
  - IPMI Command Handler
  - IPMI Command Validation Handler
  - Minimum Privilege Level
  - Command Processing Component Handler (NULL if processed by the transport itself)
  - Minimum Command Request Size
  - Maximum Command Request Size
  - Minimum Command Response Size
  - Maximum Command Response Size
  - Valid Power States (S0, S5, Both)

IPMI Standards – Channels

Channel Name	Channel Number
IPMB	0h
LAN Channel – 0	1h
LAN Channel – 1	2h
IPMI over Serial	3h
IPMI – TMODE	4h
MCTP	5h…
System Interface	Fh

All standard features for each channel is available for configuration
Standard Payloads are supported including OEM payloads

IPMI Standards - Users

- Maximum Users can be configured
- Special Username that can be used as functional enabling keys are supported
- NULL username is not supported except for the basic standard compliance

IPMI Standards – Messaging Support

- Provides the Messaging support commands for System Interface including a configurable Receive buffer

IPMI Standards - Chassis

- Supports Chassis Control, Reset and Identify commands
- Support for Front Panel Commands
- Support for Power Restore Policy
- Support for Power Cycle Interval
- Support for System Restart Cause
- Support for System Boot Options

IPMI Standards – Events

- Support for Event Receiver
- Support for Platform Events
- Configurable events for each sensors

IPMI Standards – Sensors

- Standardized Sensor Number Assignments
- Follow the Sensor Data Record (SDR) for query
- Sensor Records are stored at the platform level configured by the Platform FDK

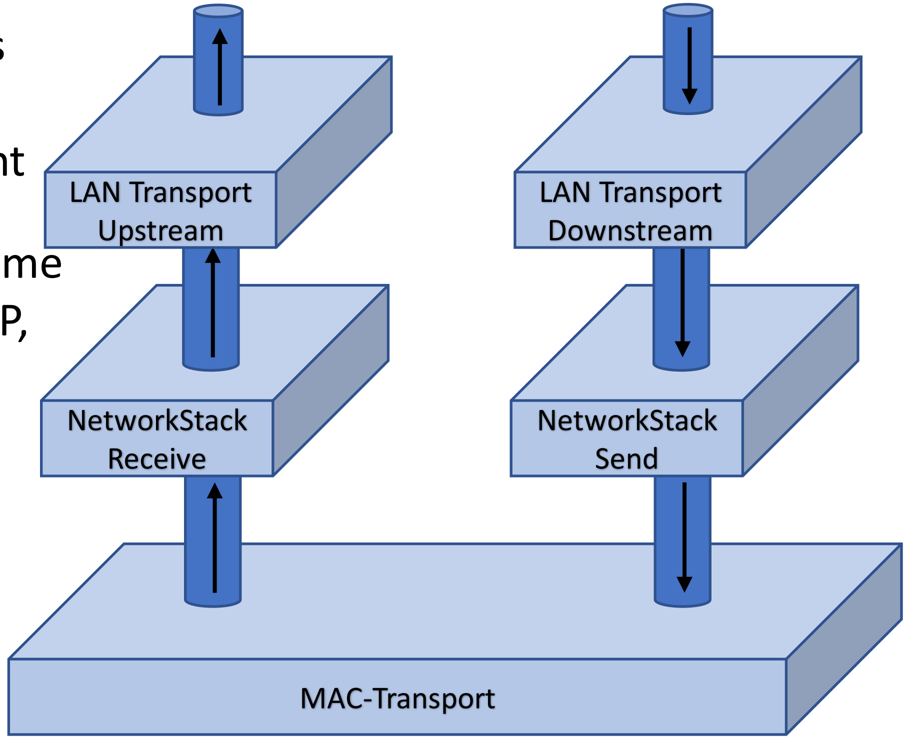
IPMI Standards – Storage

- IPMI SEL can be configured to any size with a rollback or rollover capability as described in DCMI 1.5 specification
- Sensor Event Log is a run-time component that can accept service messages for receiving SEL messages from Sensor Management and other components
- IPMI FRU is supported as per the IPMI Specification

IPMI Standards - LAN Transport

LAN Transport Upstream: Runtime component provides RMCP+ upstream interface including session management

Network Stack Receive: Runtime component provides ARP, ICMP, UDP (DHCP, Port 623)



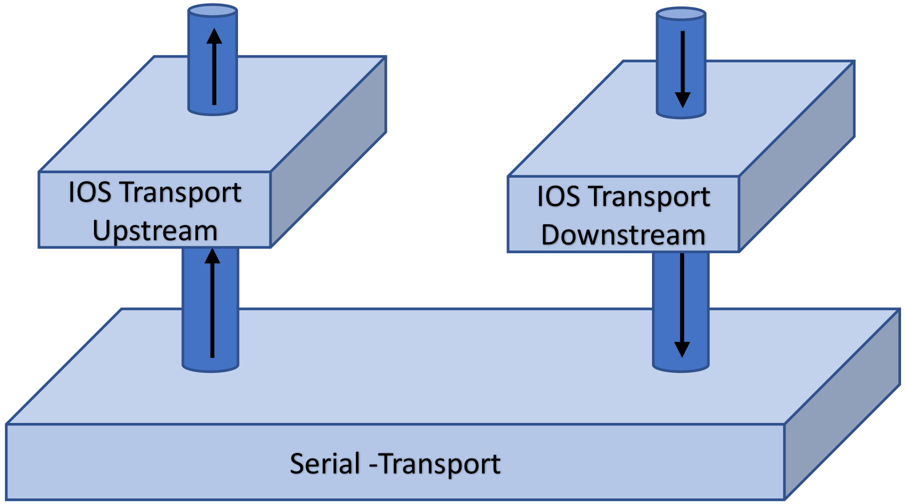
LAN Transport Downstream: Runtime component provides RMCP+ downstream interface including session management

Network Stack Send : Runtime component provides responses for ARP, ICMP, UDP (DHCP, Port 623)

MAC Transport: Provides ingress validation and control of all the packets and routing to the underlying BMC Chip interface

IPMI Standards – Basic Serial Transport

IOS Transport Upstream: Runtime Component for IPMI over Serial Session Management and transport interface



IOS Transport Downstream: Runtime Component for IPMI over Serial Session Management and transport interface

Serial Transport: Provides ingress validation and control of all the packets and routing to the underlying BMC Chip interface

IPMI Standards – Serial Over LAN

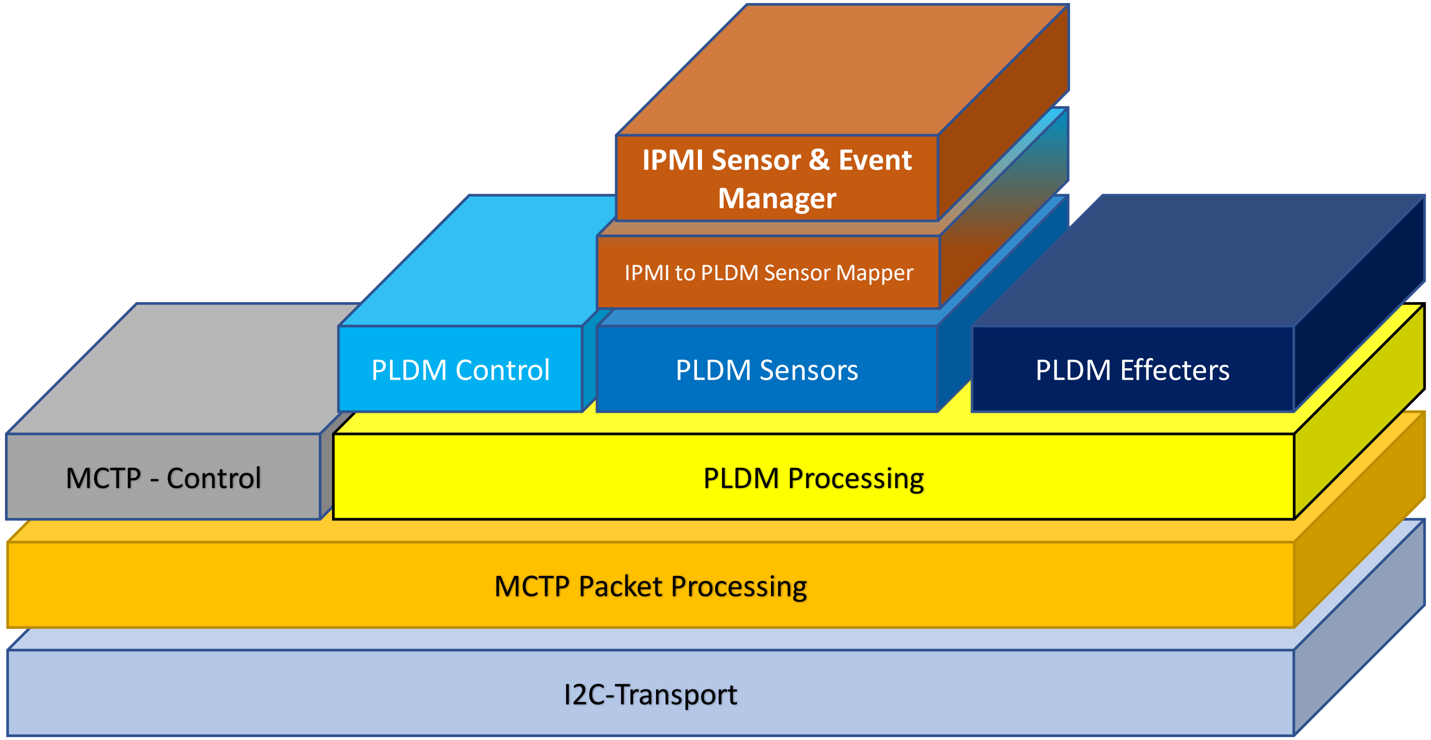
- Utilizes configurable latency buffers for both directions to match the Serial to LAN traffic

DMTF Standards – MCTP over SMBus

Allows MCTP packets processing over SMBus

Provides the MCTP ID and control

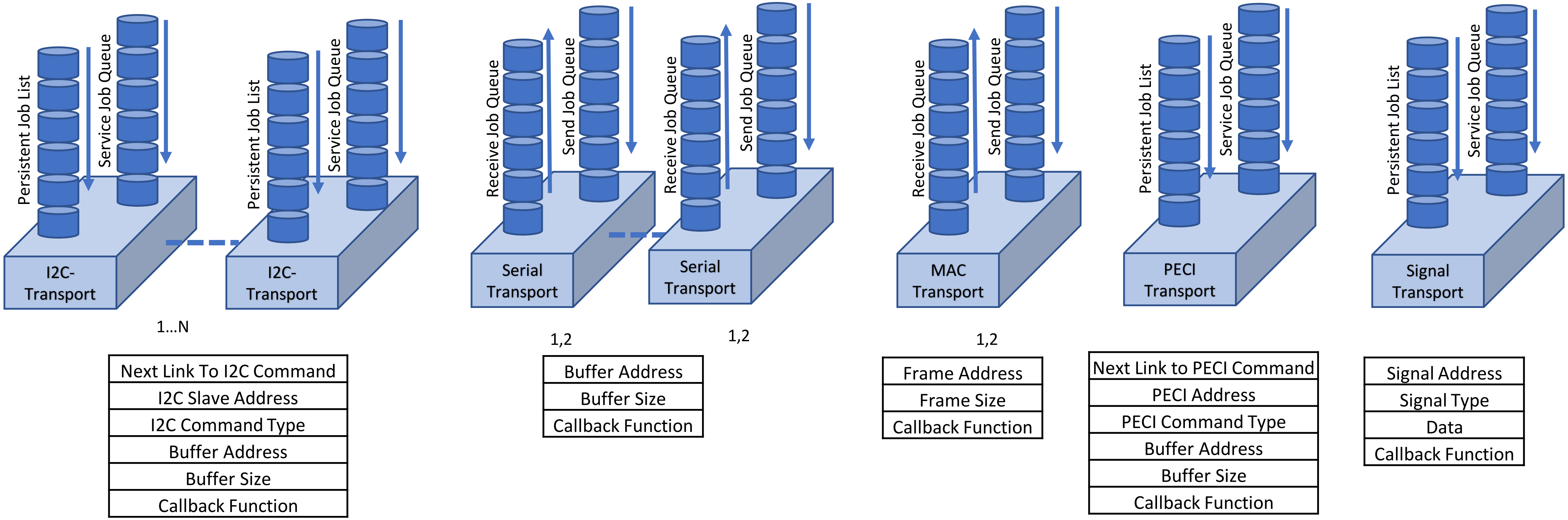
DMTF MCTP & PLDM Command Processing



OPEN BMC Firmware Engine

Transports

Transports



Next Link To I2C Command
I2C Slave Address
I2C Command Type
Buffer Address
Buffer Size
Callback Function

Buffer Address
Buffer Size
Callback Function

Frame Address
Frame Size
Callback Function

Next Link to PECI Command
PECI Address
PECI Command Type
Buffer Address
Buffer Size
Callback Function

Signal Address
Signal Type
Data
Callback Function

Platform and Chip Configuration Support

❑Seamless integration with S4H Open BMC Chip Configuration FDK

❑Seamless integration with S4H Open Platform Configuration FDK

❑Seamless integration with S4H Open Firmware Extensions FDK

Thank you!