PMCI Tools Task Force

Pat Schoeller, Intel Corporation
Jim Harford, Broadcom
PMCI WG (Tools TF Co-Chairs)

Disclaimer

- The information in this presentation represents a snapshot of work in progress within the DMTF.
- This information is subject to change without notice. The standard specifications remain the normative reference for all information.
- For additional information, see the DMTF website.
- This information is a summary of the information that will appear in the specifications. See the specifications for further details.

Session Goals

- Present the charter of the PMCI WG Tools Task Force (TF)
- Summarize the PMCI Tools TF Architecture
  - Test Client (Protocol Validator)
  - Test Service (Connector to MCTP process)
- Introduction to Python Scapy
- Show tool sample output
- Call for participation

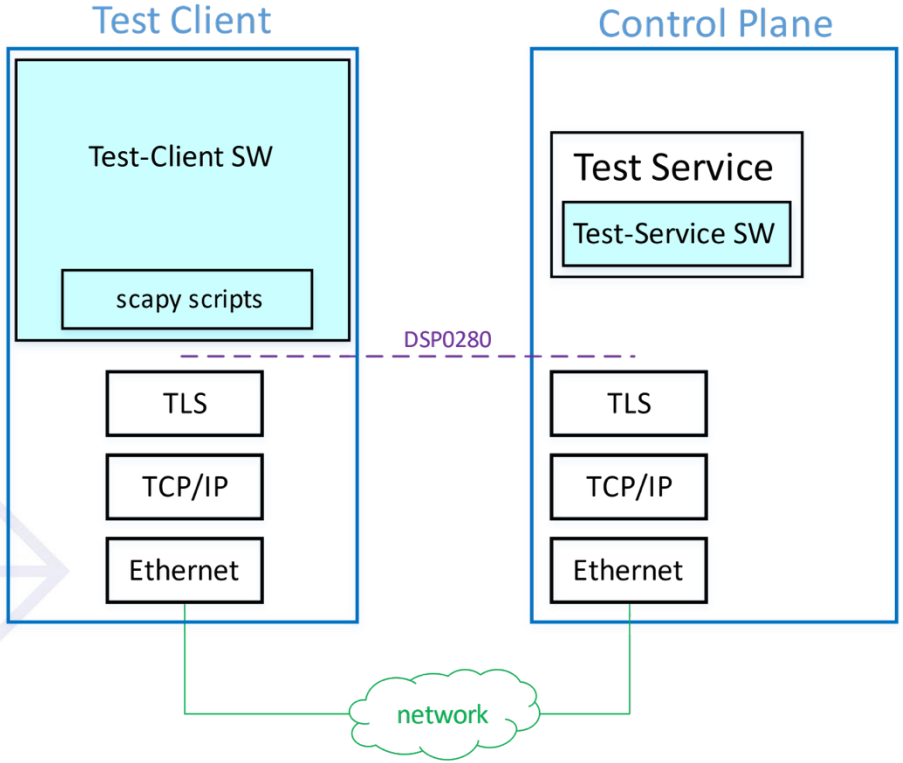
Overview of PMCI Tools Task Force

- Developing Open-Source tools to enable validation of PMCI Protocols on managed devices (adapters).
- The tools are broken down into:
  - Test Client
    - PMCI Test Tools Interface and Design Specification (DSP0280) implementation
    - PMCI Protocol Validator (Python with Scapy Library)
    - PMCI WG Protocols: PLDM Monitor & Control, NC-SI
    - Allows easy adoption of other manageability protocols transported over MCTP
    - Open-Source Code Delivery through GitHub
  - Test Service
    - Is part of the Target (e.g., BMC) Control Plane
    - Goal is to achieve 80 - 95% common code (e.g., Control Plane compiles the code into its environment).
      - Defined Test Service / Control API to encourage “Accept As Is” adoption

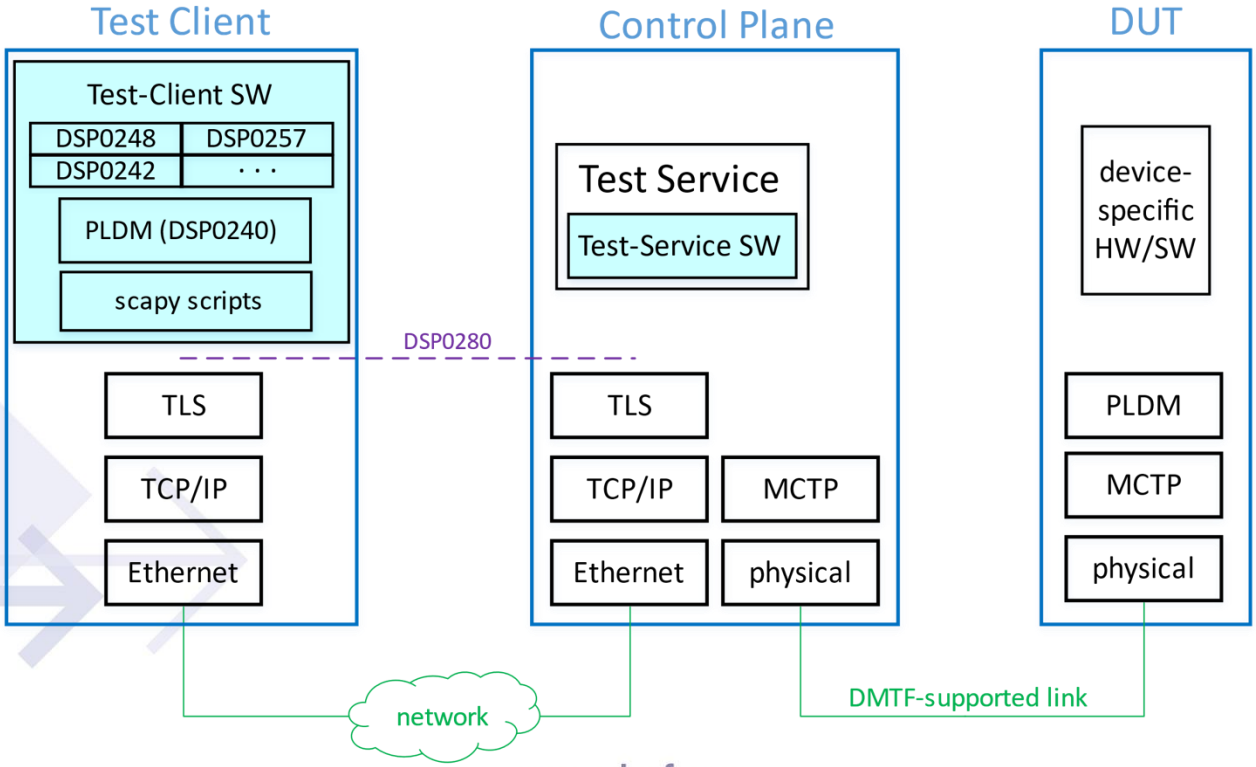
Key Features of the PMCI Tools TF deliverables

- Universal (open source with expected 80-95% compiled compatibility)
- Statically linked or stand-alone DLL style library
- Allows for diversity in BMC (Control Plane) integrations; Designed to allow BMC operations to co-exist with the Test Service Library PMCI transactions
- PMCI-Protocol-Validator provides consistent stimulus & verification of PMCI WG protocols on a device, using the Control Plane as a pass-through support agent

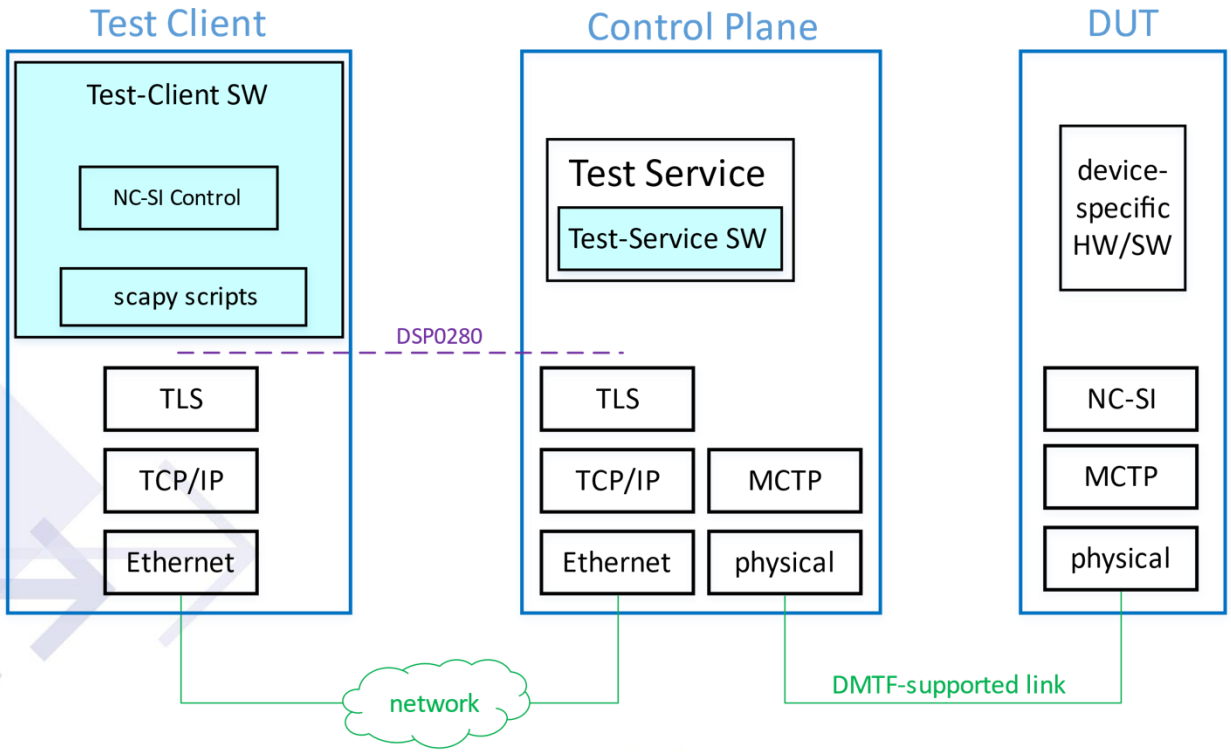
Test Control via Admin and Vendor Defined Admin Messages



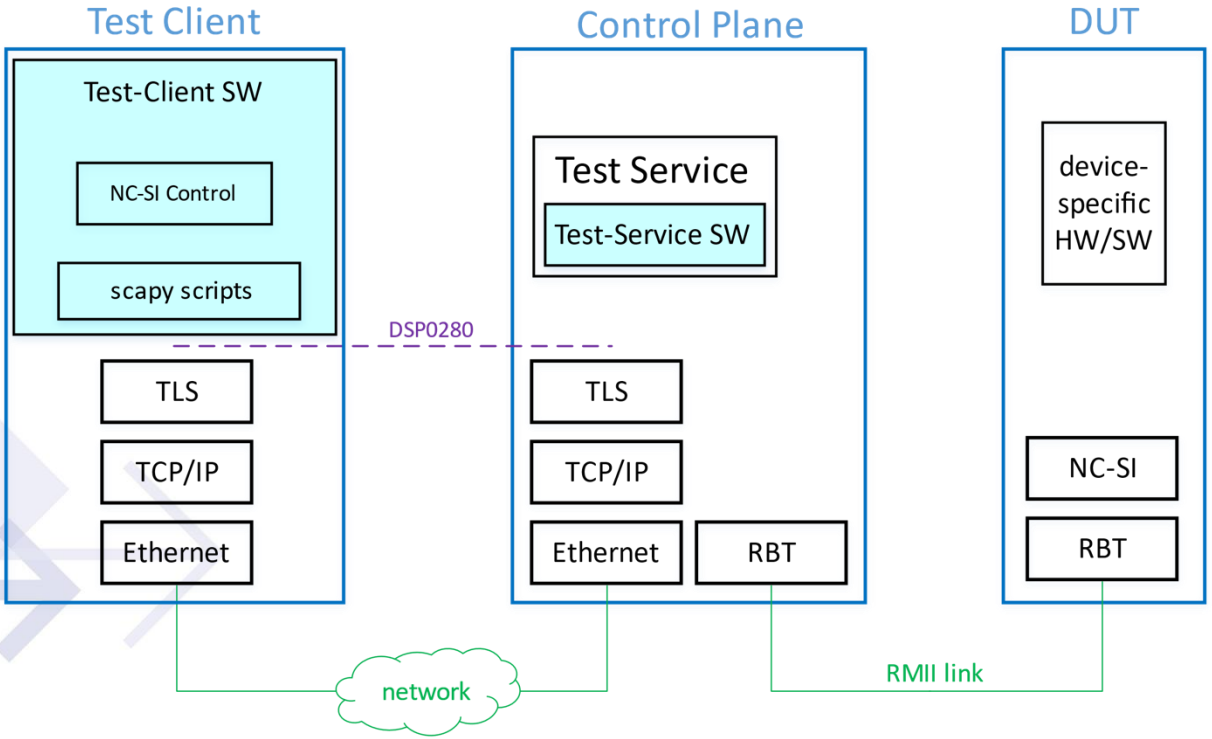
Testing PLDM Functionality



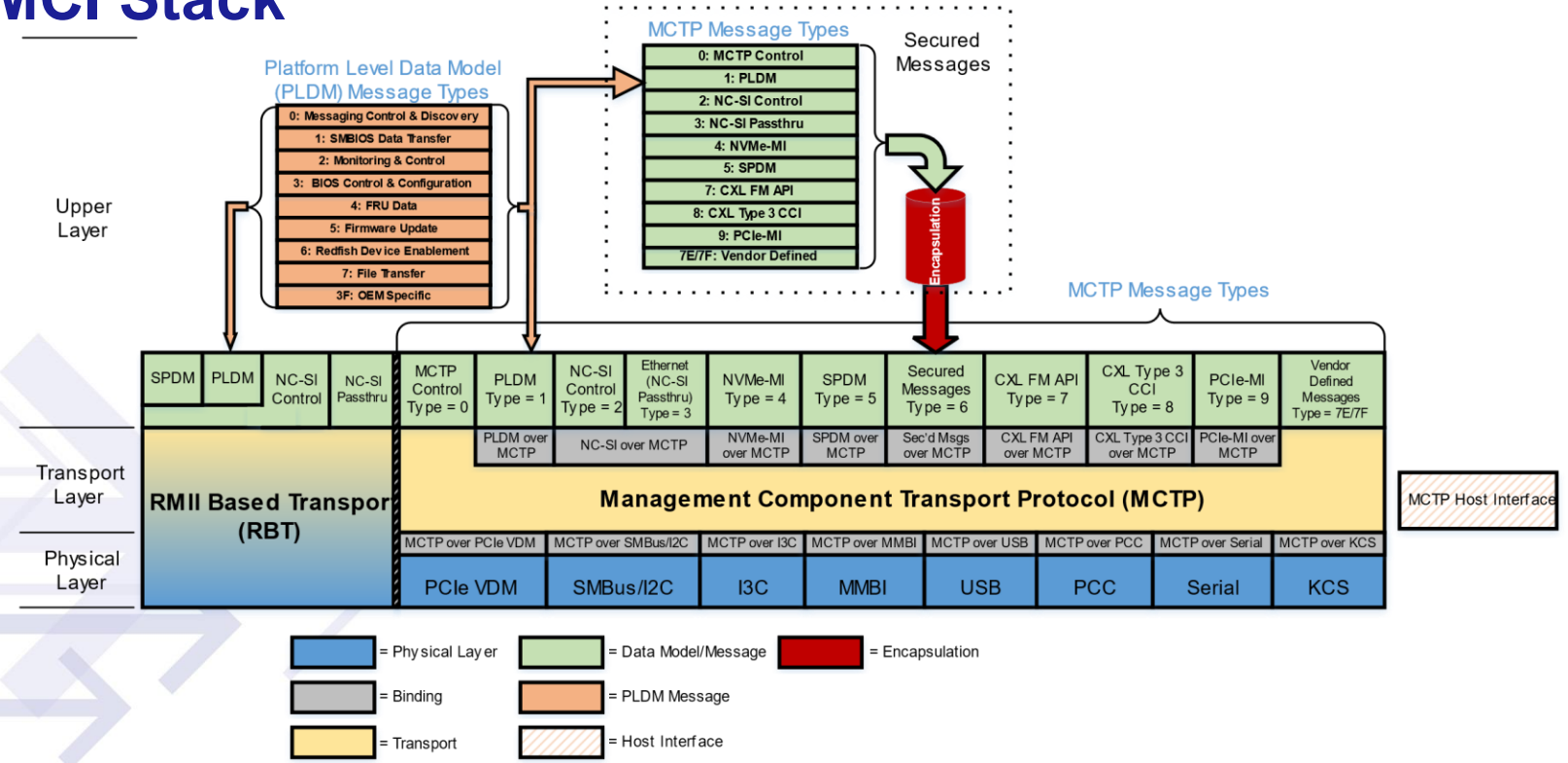
Testing NC-SI Over MCTP Functionality



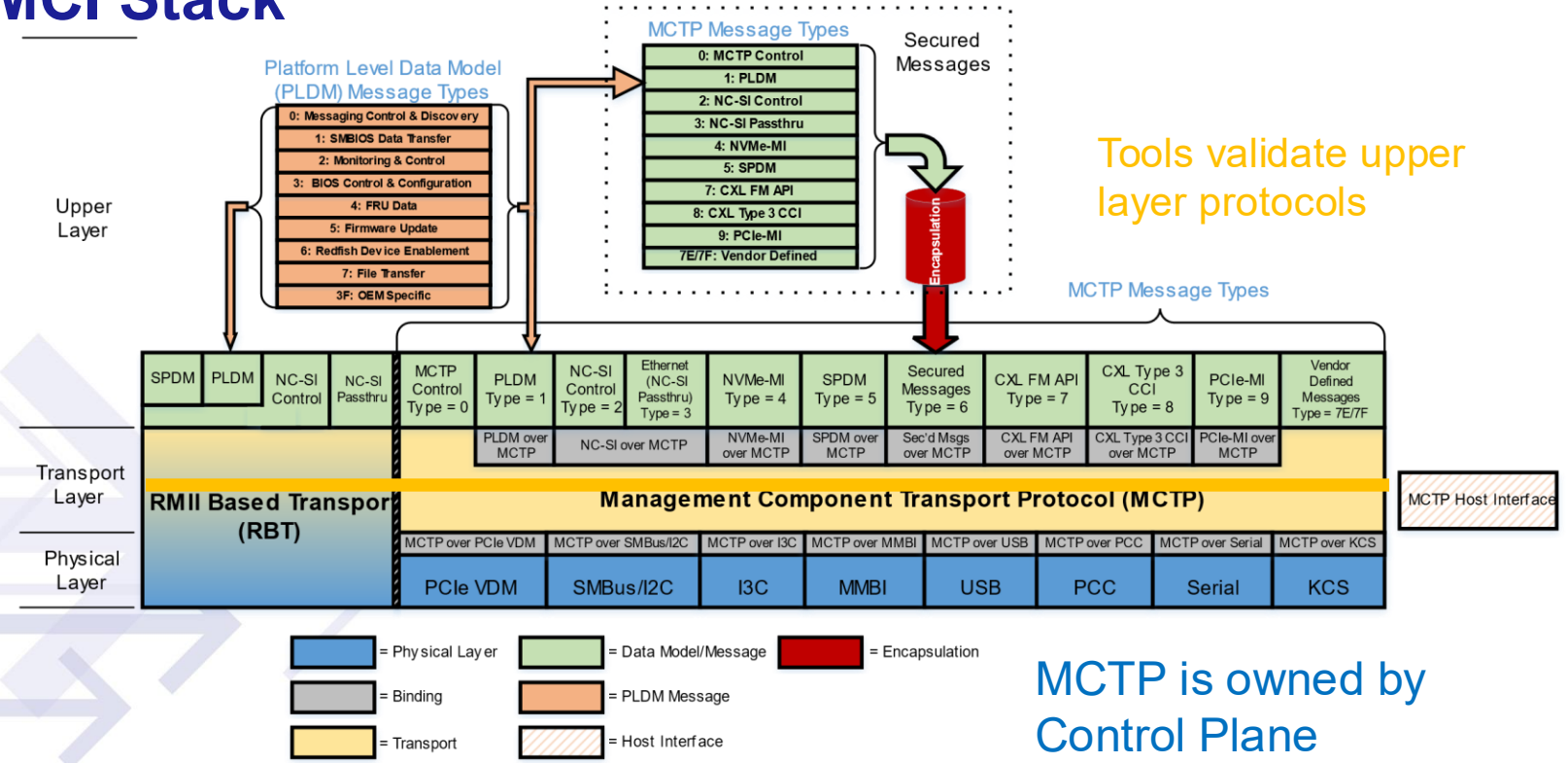
Testing NC-SI Over RBT Functionality



PMCI Stack


Platform Level Data Model (PLDM) Message Types
0: Messaging Control & Discovery
1: SMBIOS Data Transfer
2: Monitoring & Control
3: BIOS Control & Configuration
4: FRU Data
5: Firmware Update
6: Redfish Device Enablement
7: File Transfer
3F: OEM Specific
MCTP Message Types
0: MCTP Control
1: PLDM
2: NC-SI Control
3: NC-SI Passthru
4: NVMe-MI
5: SPDM
7: CXL FM API
8: CXL Type 3 CCI
9: PCIe-MI
7E/7F: Vendor Defined
Secured Messages
Encapsulation
MCTP Message Types
Upper Layer
SPDM
PLDM
NC-SI Control
NC-SI Passthru
MCTP Control Type = 0
PLDM Type = 1
NC-SI Control Type = 2
Ethernet (NC-SI Passthru) Type = 3
NVMe-MI Type = 4
SPDM Type = 5
Secured Messages Type = 6
CXL FM API Type = 7
CXL Type 3 CCI Type = 8
PCIe-MI Type = 9
Vendor Defined Messages Type = 7E/7F
Transport Layer
PLDM over MCTP
NC-SI over MCTP
NVMe-MI over MCTP
SPDM over MCTP
Sec'd Msgs over MCTP
CXL FM API over MCTP
CXL Type 3 CCI over MCTP
PCIe-MI over MCTP
RMII Based Transport (RBT)
Management Component Transport Protocol (MCTP)
MCTP Host Interface
Physical Layer
MCTP over PCIe VDM
MCTP over SMBus/I2C
MCTP over I3C
MCTP over MMBI
MCTP over USB
MCTP over PCC
MCTP over Serial
MCTP over KCS
PCIe VDM
SMBus/I2C
I3C
MMBI
USB
PCC
Serial
KCS
= Physical Layer
= Data Model/Message
= Encapsulation
= Binding
= PLDM Message
= Transport
= Host Interface

PMCI Stack



Tools validate upper layer protocols

MCTP is owned by Control Plane

Scapy Overview

- Scapy is a powerful interactive packet manipulation library written in Python. Scapy can encode / decode packets of a wide number of protocols
- Python Library that outputs “Human Readable” but machine parsing enabled
- Test Client is architected as Python with Scapy to allow individual innovation without compilation
- Python has broad support in the industry

Scapy Connect Request & Response Decode

```
###[ PTTI Test Service Wrapper ]###
  Version    = 0x10
  ProtocolType= PTTI Admin
  Reserved   = 0
  Direction = TC to TS Request
  TestClientID= 0x0aaa
###[ PTTI Connect Request ]###
     CommandCode= Connect
     SecurityParameterLength= 6
     SecurityParameter= [0x31, 0x32,
0x33, 0x34, 0x35, 0x36]
```

```
###[ PTTI Test Service Wrapper ]###
  Version    = 0x10
  ProtocolType= PTTI Admin
  Reserved   = 0
  Direction = TS to TC Response
  TestClientID= 0x0aaa
###[ PTTI Connect Response ]###
     CommandCode= Connect
     ResponseCode= SUCCESS
     TestServiceVersion= 0x10
     TestClientID= 0xdeadbeef
```

Goals for 2025

- Complete the Test Service Code and Open Source by 4Q2025
  - Follows the SPDM Tools GitHub Publishing Model
  - Publish a Control Plane integration example
- Complete the Test Client Code and Open Source by 4Q2025
  - Base PLDM DSP0240 / DSP0248 functionality
  - Base NC-SI functionality
  - Modular design allows open contributions for other MCTP enabled manageability protocols (such as NVMe MI, PCIe MI, etc)

Example Tools GitHub Repository Structure (Subject to change)

- https://github.com/DMTF/lib-test-service
  - examples/ - mock Control Plane using libTestService (implements interfaces/)
  - interfaces/ - .h file documenting interfaces that the control plane must provide
  - src/ - ANSI C source for a Test Service
- https://github.com/DMTF/PMCI-Protocol-Validator
  - src/ - python source code for validating PLDM/NC-SI
    - /dsp0240
    - /dsp0248
    - /dsp0267
    - /dsp0218
    - /dsp0222

For more information,
visit dmtf.org

Learn about the PMCI working group at
dmtf.org/standards/pmci

Thank you!