



# SERVICE AVAILABILITY™ FORUM

## Platform Management 101

HPI and how it touches AIS....

*Portable Platform Management for High  
Availability Systems*

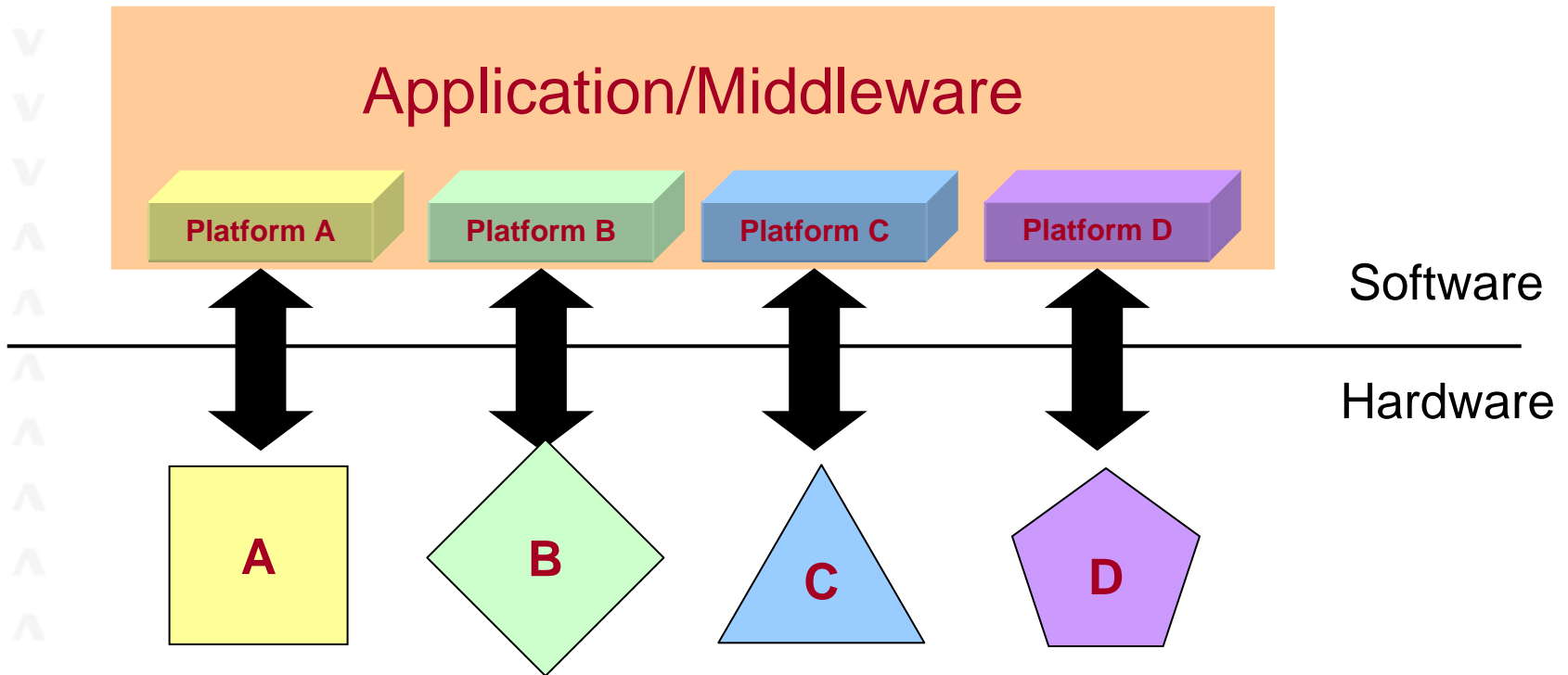
# Agenda

- ***High level overview of Hardware Platform Interface***
- ***Where HPI and AIS High Availability platforms are rich in platform management capabilities***
- ***Monitoring and controlling platform is essential to continuous service availability***
- ***Platform differences lead to non-portable applications***
- ***HPI normalizes the platform view for clients***
- ***SAF AIS Middleware is just such a client***

# The Problem

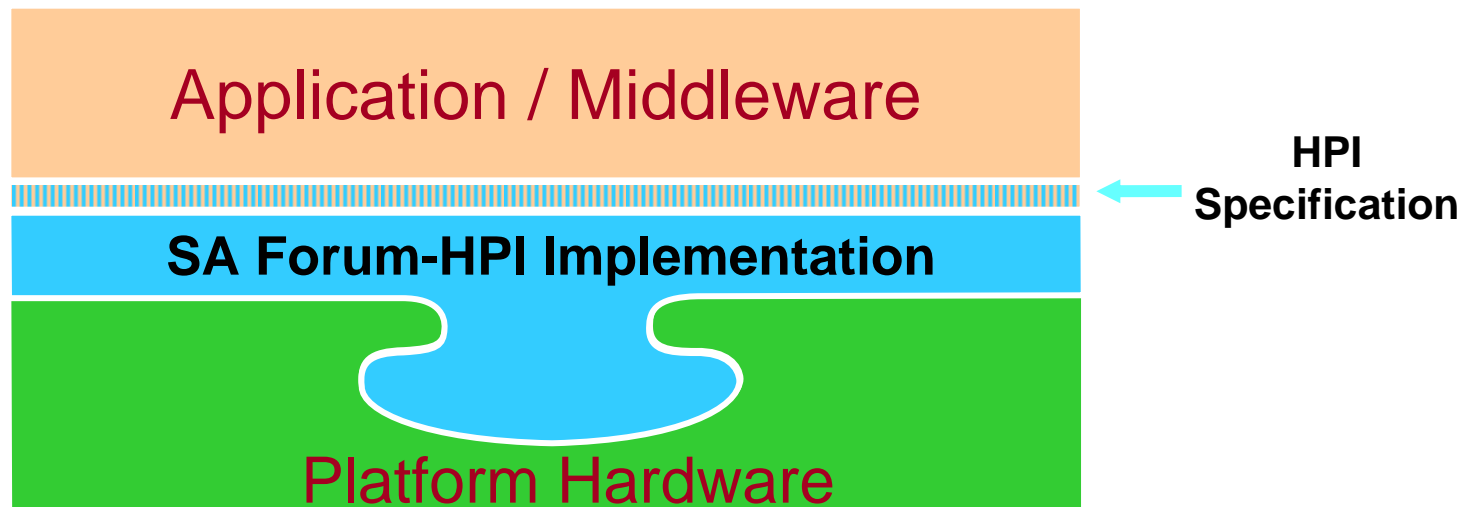
- ***High Availability platforms are rich in platform management capabilities***
- ***Monitoring and controlling platform is essential to continuous service availability***
- ***Platform differences lead to non-portable applications***

- ***Without standard interface, platform specific software is required to manage each different platform***

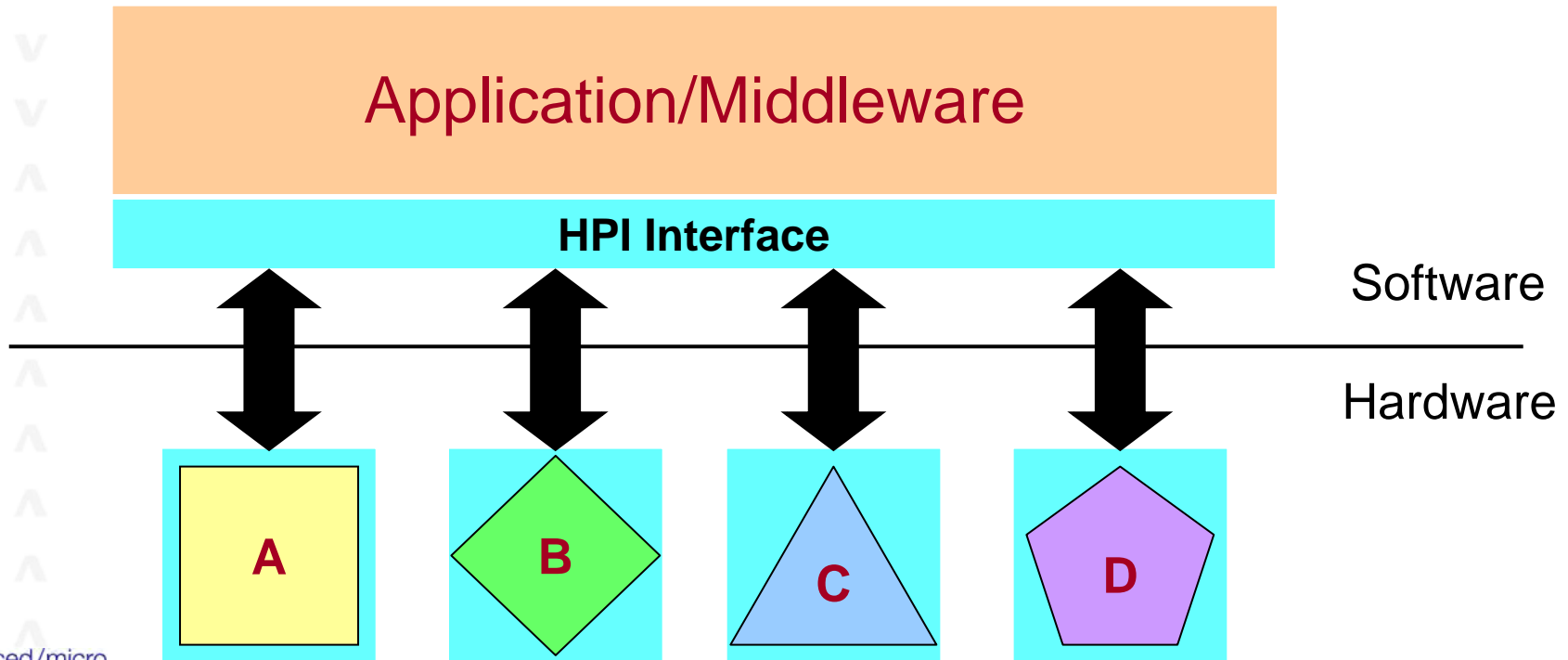


# SAF HPI Specification Value Proposition

- **SAF HPI = Service Availability Forum Hardware Platform Interface**
- **Standard interface to platform specific or proprietary features of hardware platforms**



- *HPI eliminates the burden of proprietary application/middleware interface software, allowing for management of heterogeneous platforms.*



# HPI Technical Challenges

- **Wide variety of platform architectures to be accommodated**
- **Standardization of hardware not likely (or desirable)**

CompactPCI  
1U/2U Servers  
VME  
NEBS  
PICMG 1.0  
PICMG 2.16  
microTCA  
Enterprise  
AdvancedTCA

# HPI Technical Challenges

- ***Even more platform management architectures***
- ***Area of differentiation between platform vendors***

Alarm Managers  
IPMI  
Out of Band  
Proprietary  
Dry Contact Relays  
PICMG 2.1  
In Band  
Shelf Managers

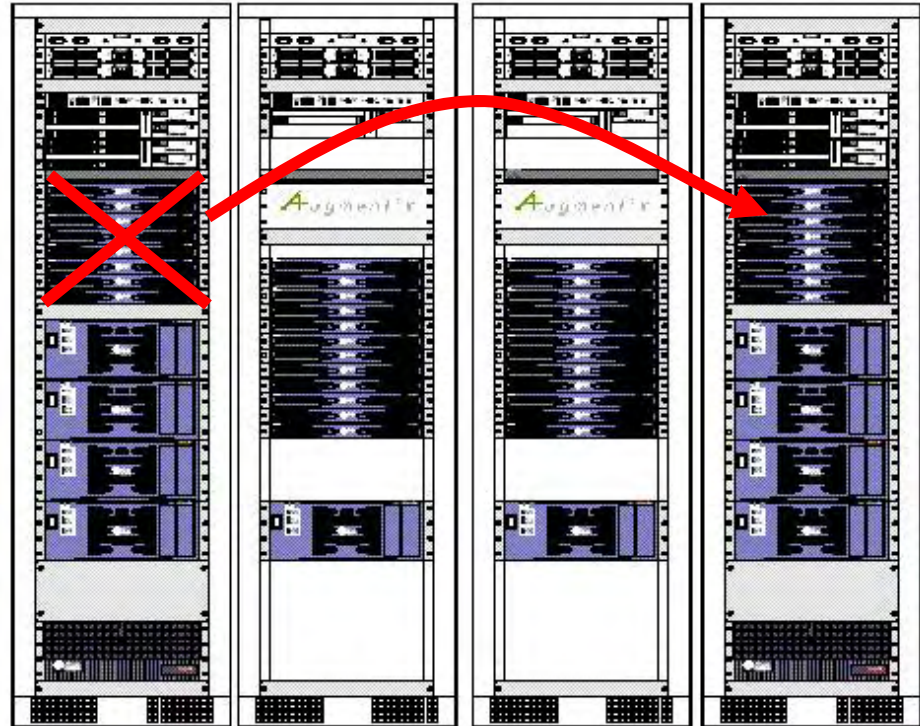
# HPI Technical Challenges

- ***“Platform” is increasingly a loosely-coupled, distributed set of compute and I/O nodes***
- ***Need centralized, common interface to distributed heterogeneous subsystems***



# HPI Technical Challenges

- ***Fault-resilience and hot swap is fundamental***
- ***Applies not only to modules providing service –***
- ***- But also to parts of the management infrastructure itself***



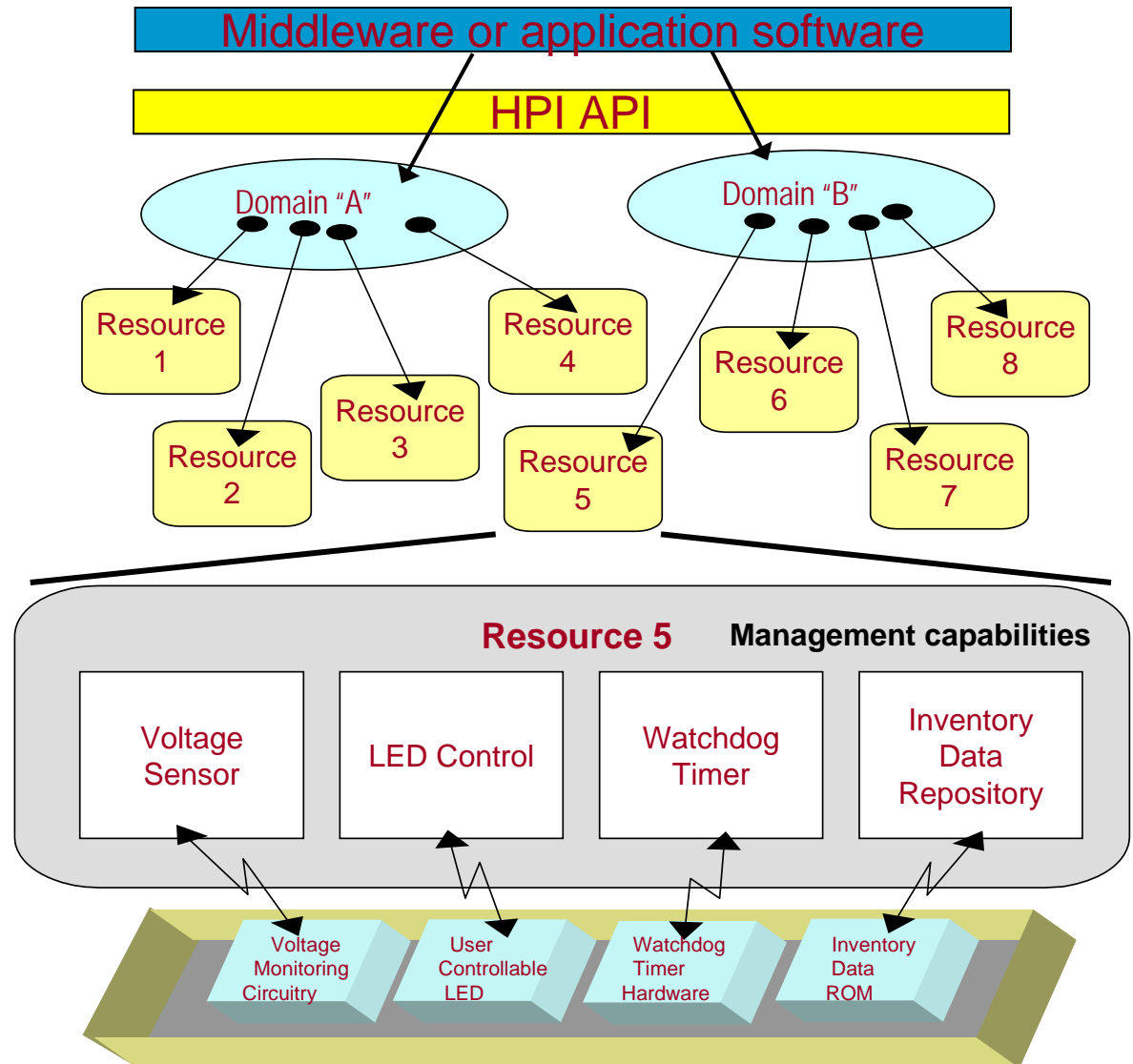
# HPI Architecture - Big Picture

Users open sessions with Domains via HPI library functions

Domains contain Resources representing parts of the platform management infrastructure

Resources have Management Capabilities accessible by users

Resource Management Capabilities map to platform management hardware



# HPI Architecture

- **Resources are contained in “Domains”**
- **Domains provide standard methods for application software to discover and access resources and their management capabilities**
- **Domain architecture allows common interface to large heterogeneous systems**
  - *Example: Separate domain for each rack or sub-rack in a platform*
- **Domain architecture allows modeling of redundant management access paths in platform**

# HPI Architecture - Domains

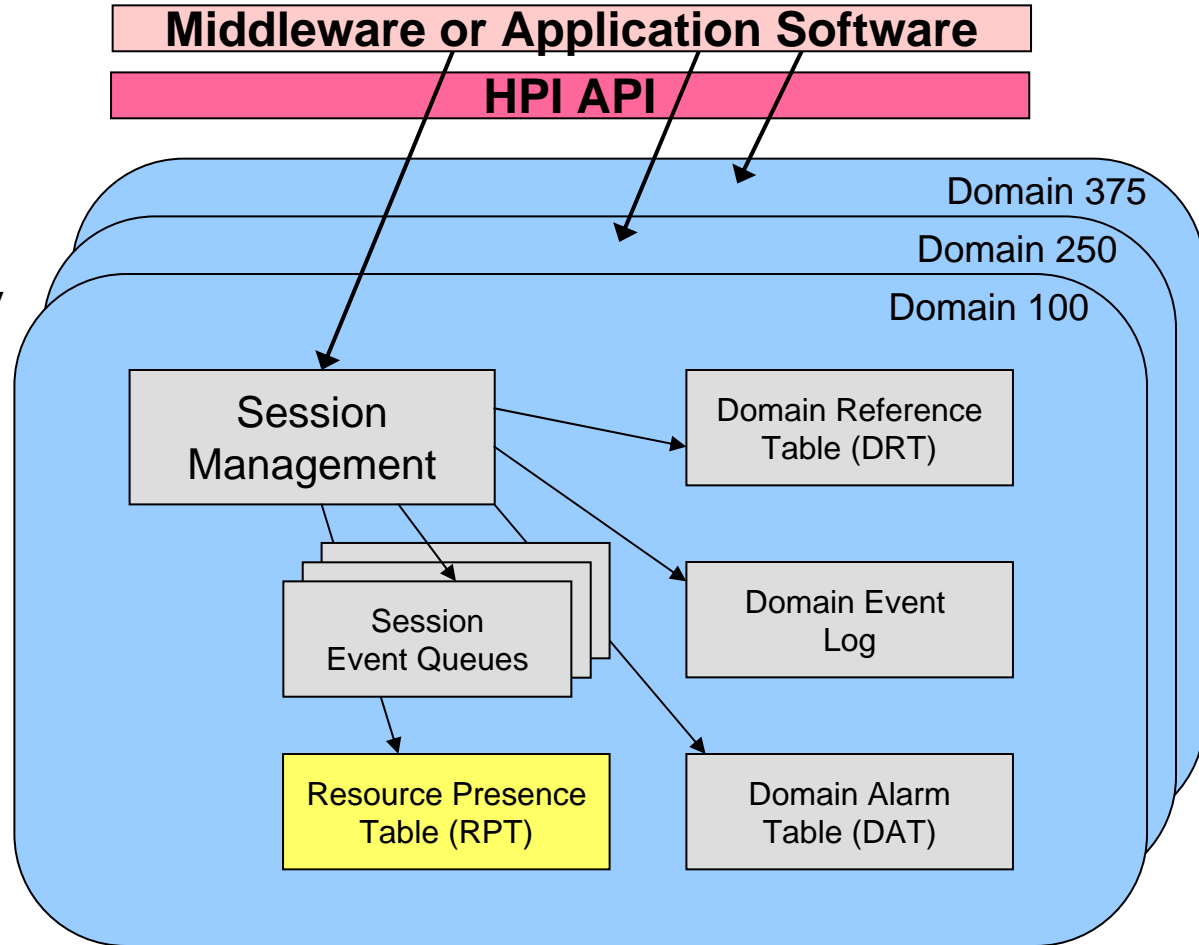
➤ Users open Sessions for **Domains** via HPI library functions

➤ **Domain** contains summary data about Resources that are contained in that Domain

- Session Event Queues
- Domain Event Log
- Resource Present Table
- Domain Alarm Table

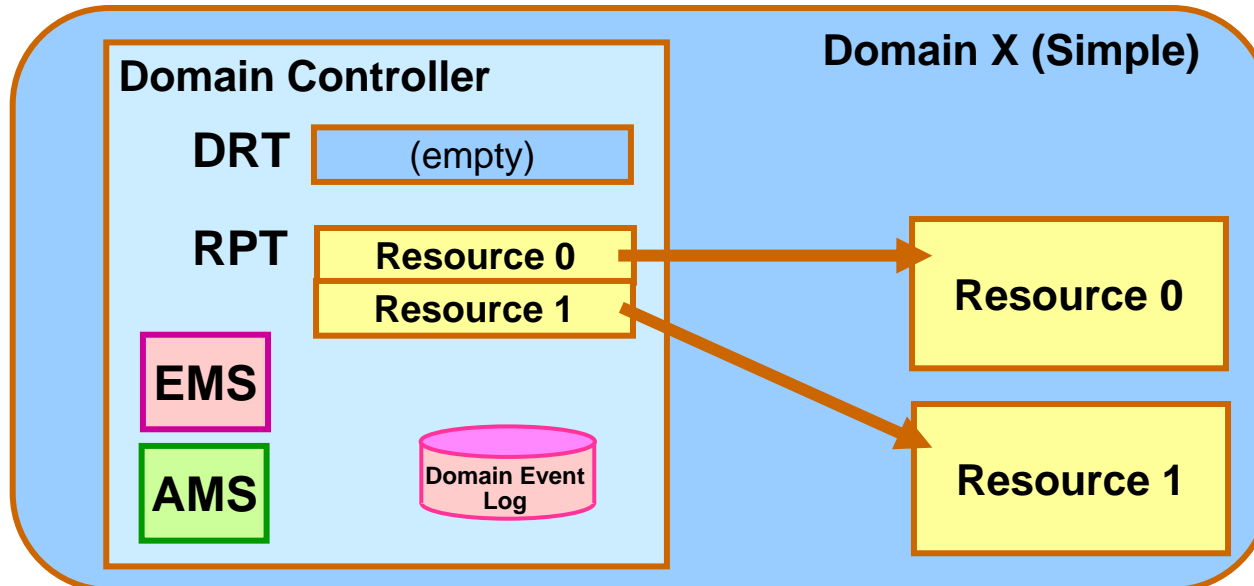
➤ **Domain** provides references to additional Domains

- Domain Reference Table



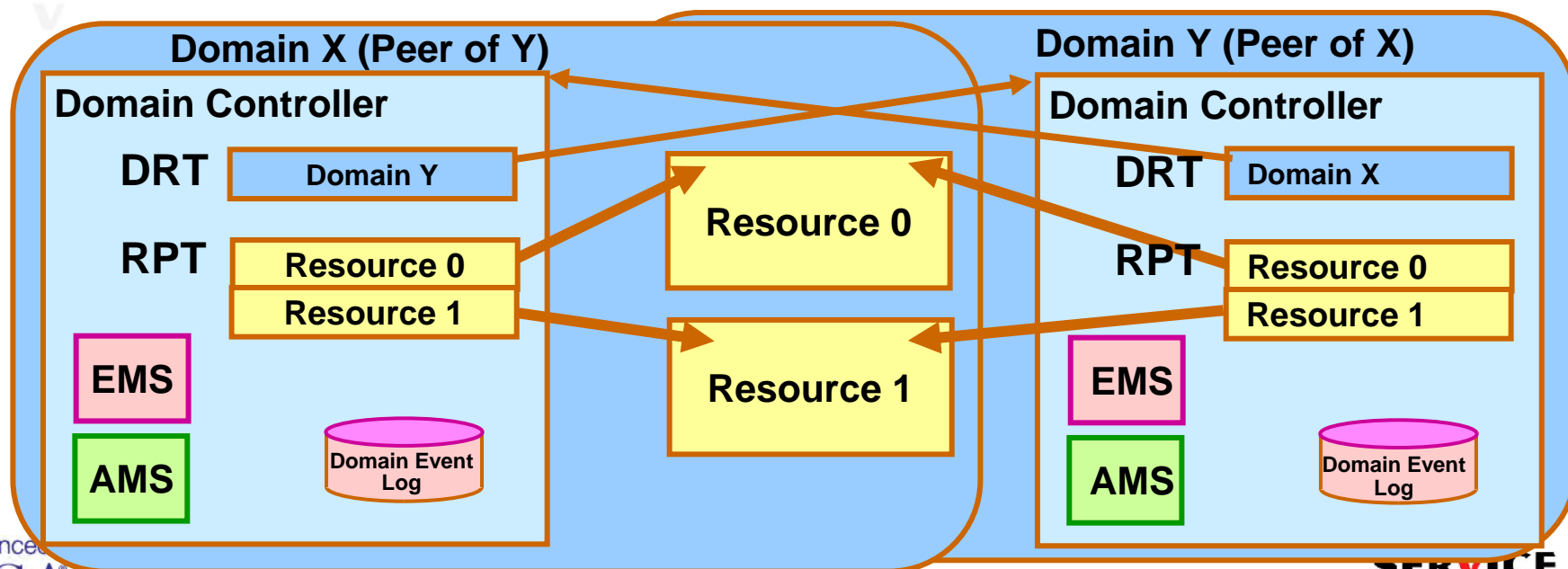
# HPI Architecture - Simple Domain

- **The Simple Domain Architecture contains of a single domain that contains only resources**
  - Only the Resource Presence Table (RPT) is populated
  - The Domain Reference Table (DRT) remains empty because the simple domain does not reference any other domain



# HPI Architecture - Peer Domains

- **The Peer Domain Architecture consists of two or more domains that contain the same resources and the same domain references**
  - Each domain contains a Resource Presence Table (RPT) that lists all resources present in the domain
  - A flag in a Domain Reference Table (DRT) entry indicates that a domain reference is for a peer domain



# HPI Architecture

➤ ***Platform management functions are represented by sets of abstract “Management Capabilities” that act on platform Entities***

➤ Sensors

➤ Controls

➤ Watchdog Timers

➤ Inventory Data  
Repositories

➤ Annunciators

➤ *Hotswap functions*

➤ *Power control functions*

➤ *Reset control functions*

➤ *Event Logs*

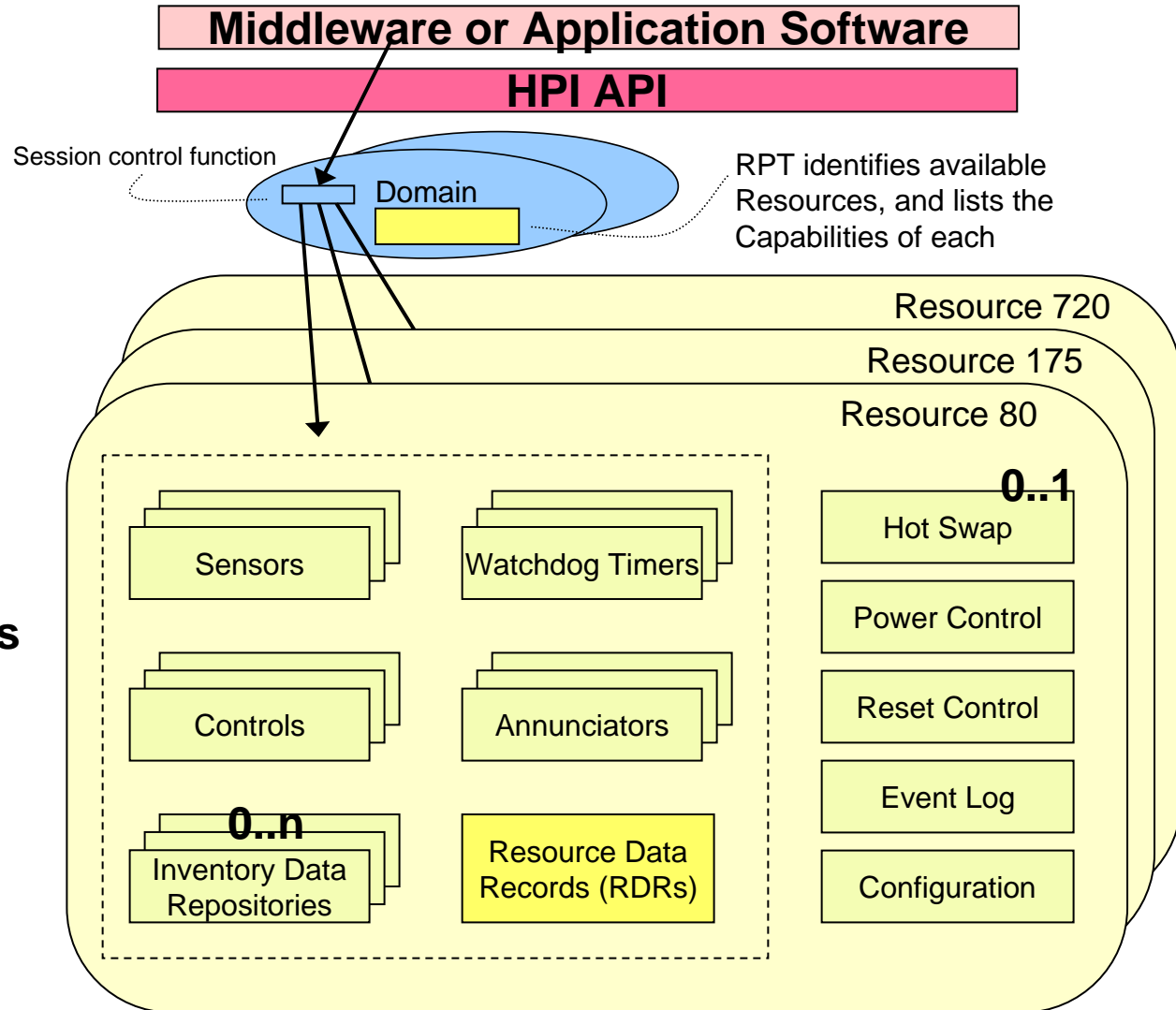
➤ *Resource Configuration*

# HPI Architecture

- **Management Capabilities are collected in “Resources”**
- **All management capabilities in a single resource share a common accessibility**
  - *Example: management functions of a single management controller in a system*
- **Resources in HPI may be added or deleted from the system over time**
  - Supports failure and hot-swap of management infrastructure pieces

# HPI Architecture - Resources

- HPI users access **resources** via sessions opened for a domain
- **Resources** contain several types of management **capabilities** identified in **RPT entries**
- Five types of capabilities are provided by **management instruments** described in **Resource Data Records**
- Five additional types of capabilities may exist on a **one-per-resource** basis



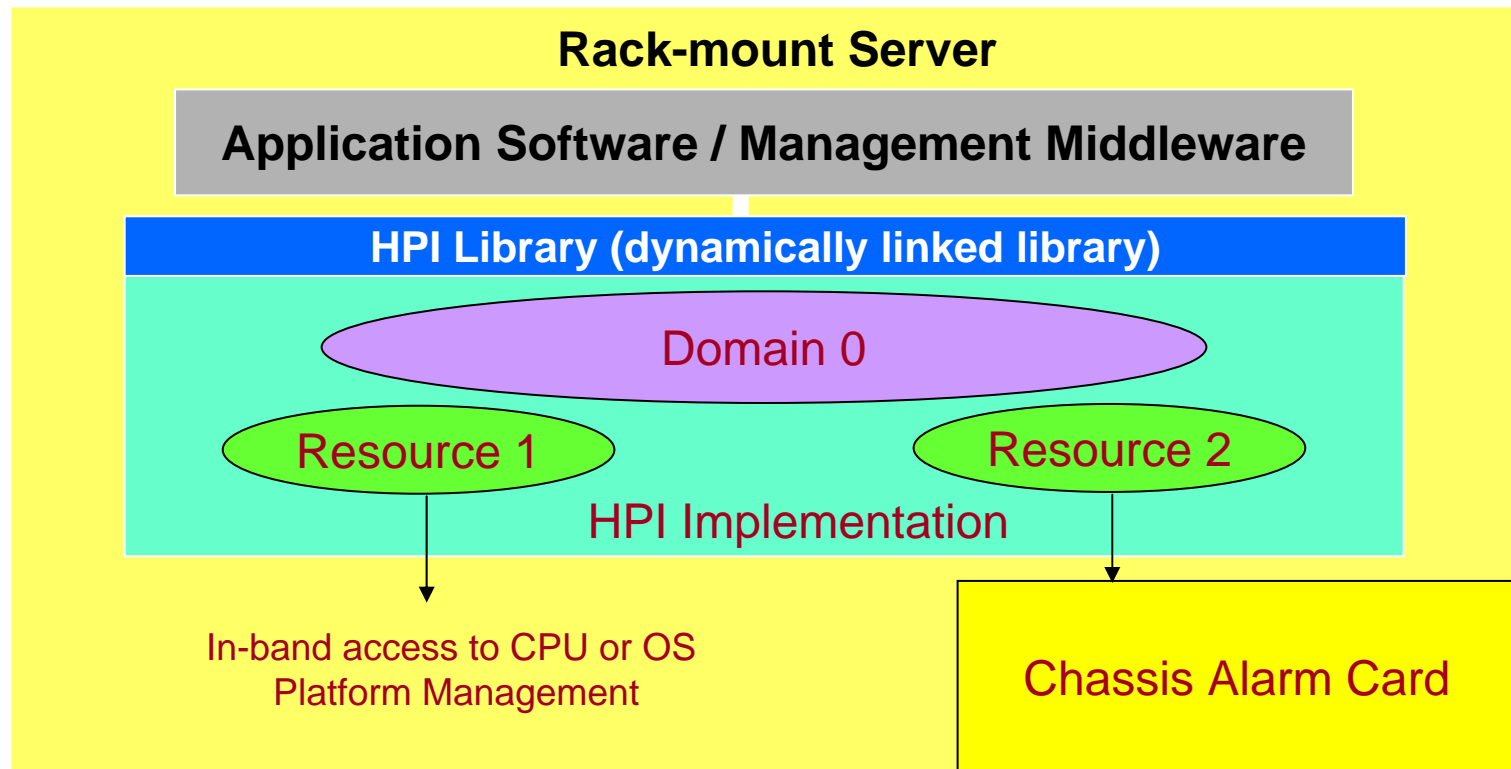
# Implementation and Use of HPI

- ***Platform vendor or system integrator provides library functions and header file that implement standard HPI API for a particular platform***
  - ....Radisys, Continuous Computing, IBM, Sun, Mercury, Kontron, and others...
- ***Application programs needing platform management access use a standard header file “SaHpi.h” and call standard C language functions***

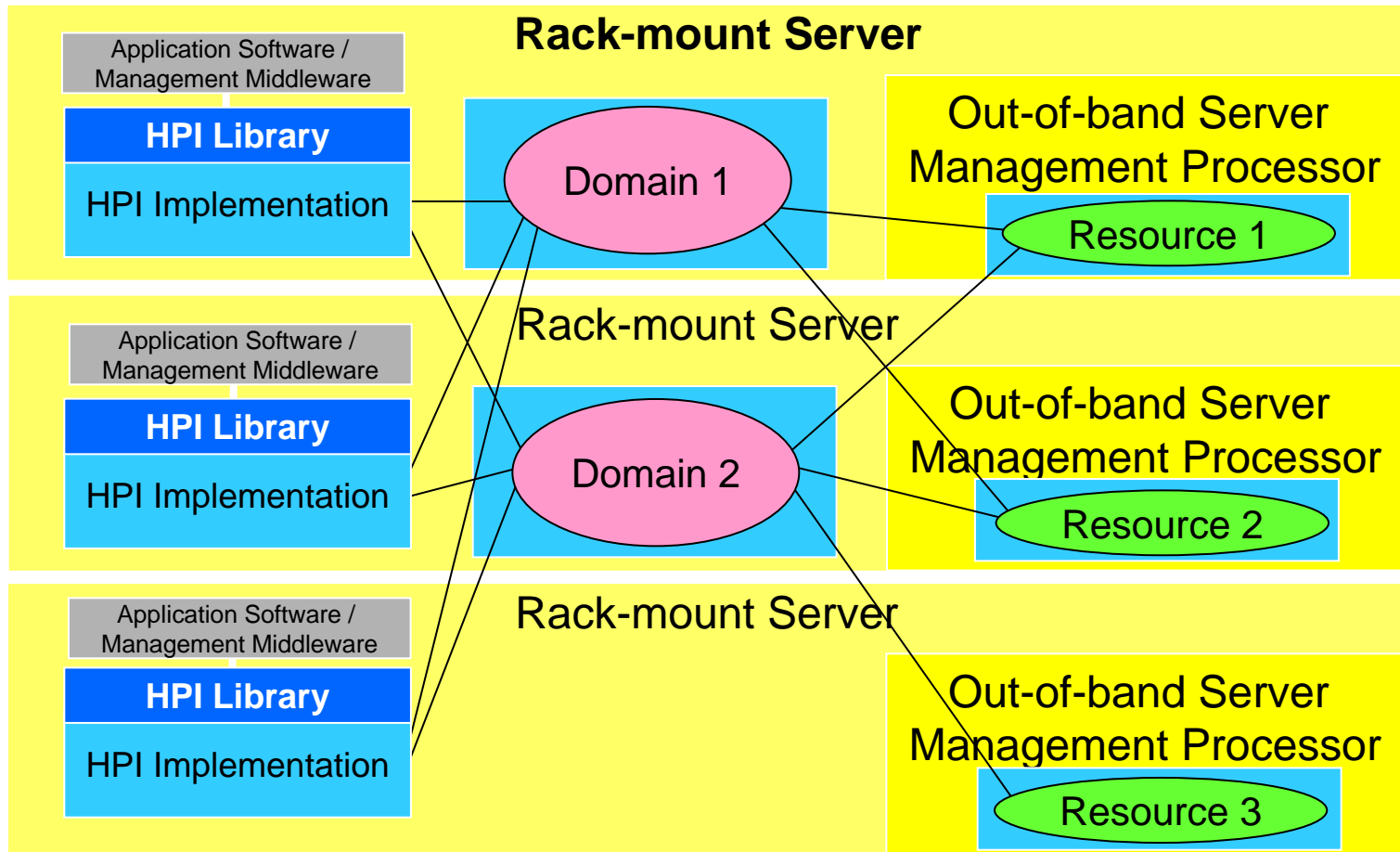
# HPI Implementation

- *The HPI standard addresses only the API library interfaces – not how the library is implemented*
- *HPI functionality is most useful in a distributed environment including hot-swappable redundant modules*
- *HPI library often is just a “front-end” for some distributed platform management infrastructure (ex. Intelligent Platform Management... IPMI, IPMB, IPMC..)*

# Example HPI Implementation Single Node

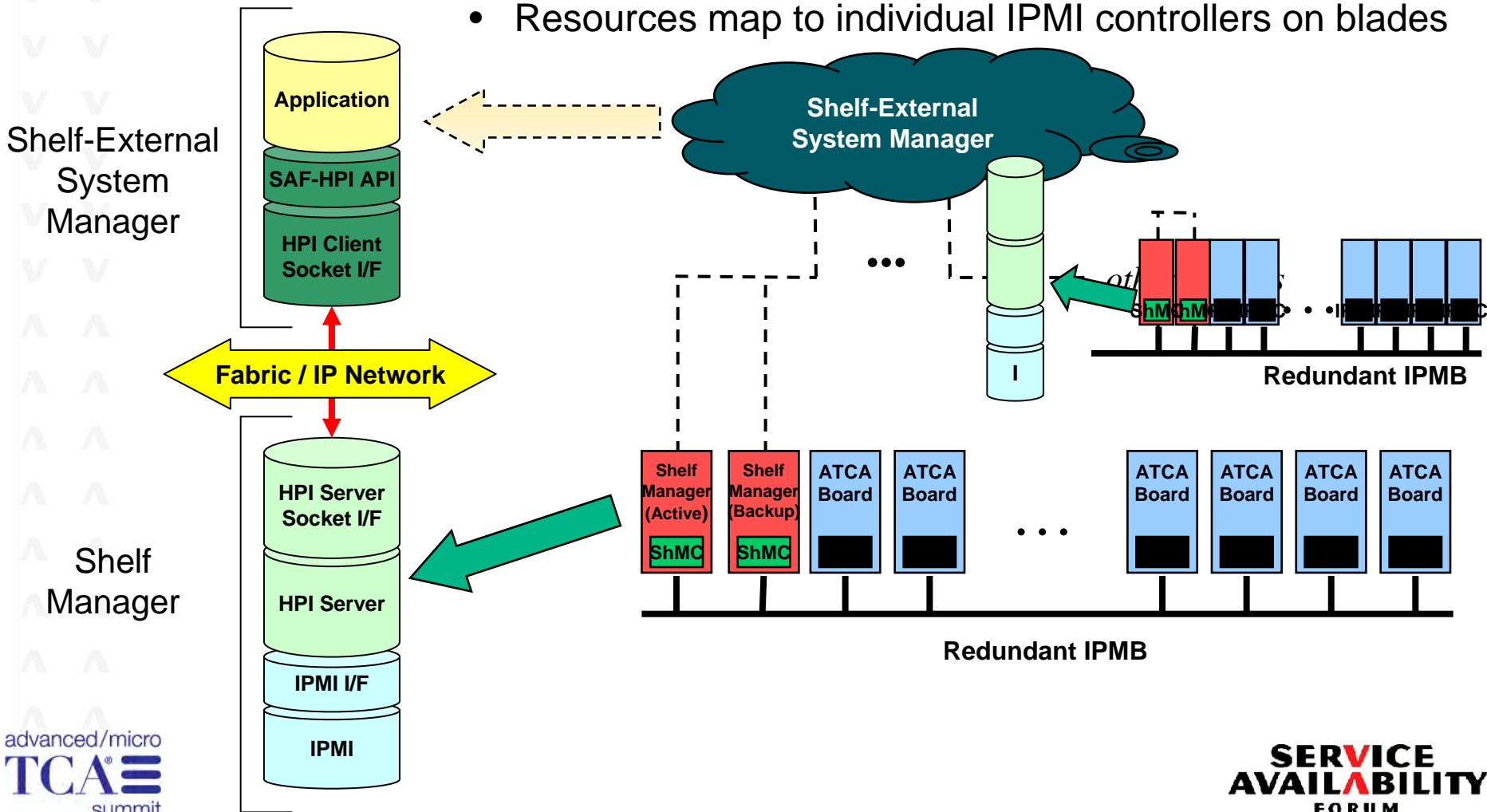


# Example HPI Implementation Multi-Node



# Example HPI implementation in an AdvancedTCA system

- User library implements client side of client/server architecture communicating to shelf manager
- Shelf managers provide domain functions
- Resources map to individual IPMI controllers on blades



# AIS Middleware as HPI Client

- ***HPI and AIS Functionality collaborate***
  - Platform Management (PLM)
  - Software Management Framework (SMF)
  - Diagnostic Management Framework (DMF)
- ***Use case: HPI as service provider to HA Middleware***
  - HPI Access must have no single point of failure
- ***Platform Management (PLM) fills a gap***
  - HW to SW dependencies and the Information Model

## HPI and AIS Functional collaboration

- Some AIS specifications are real, some only hoped for...

### ***AIS Software Management Framework (SMF) R5, A.01.01***

- A service and framework that describes how to construct and execute an upgrade ‘campaign’, such as an in-service rolling upgrade

### ***AIS Platform Management (PLM) R6, A.01.01***

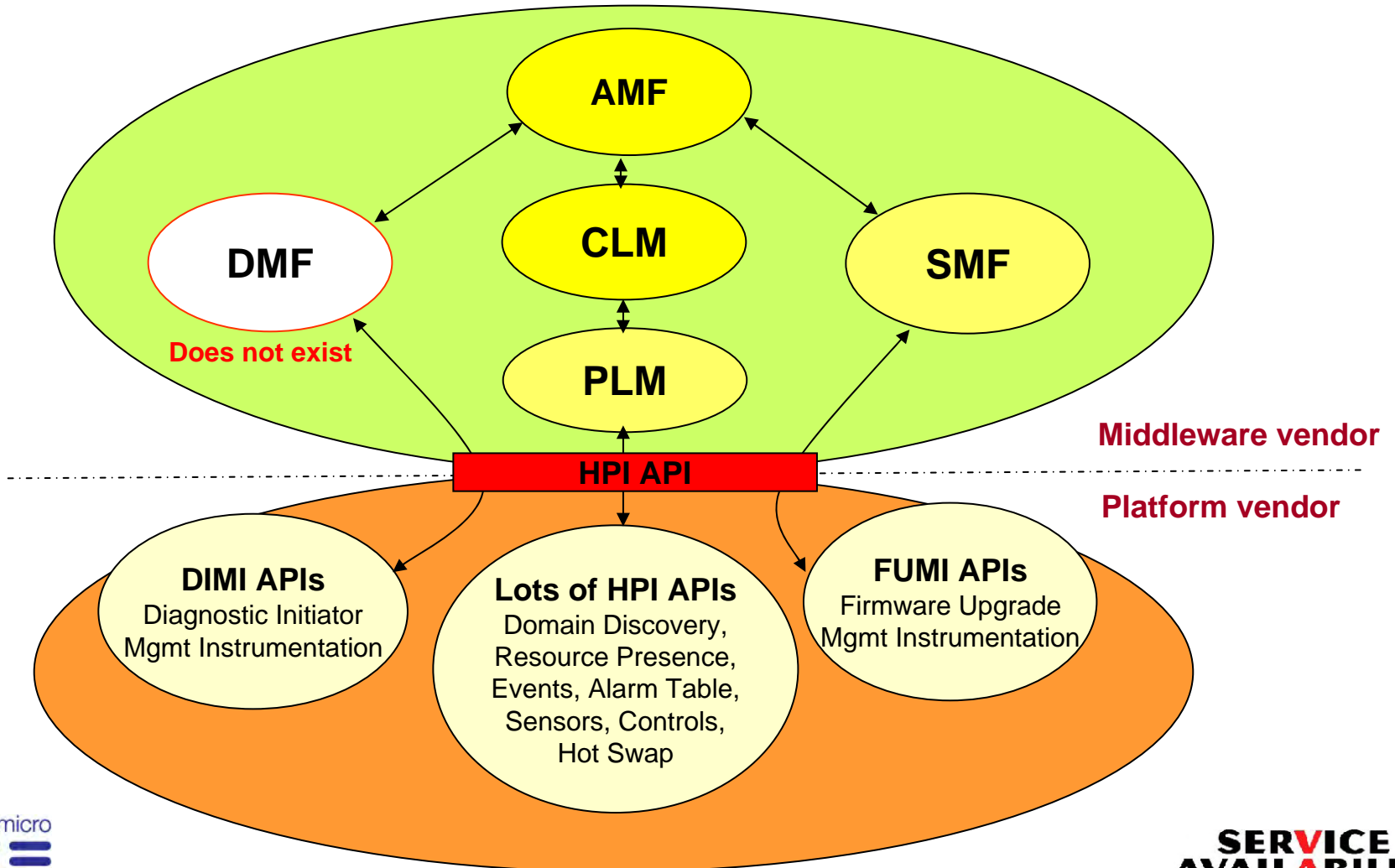
- Provides a conceptual and semantic bridge between the HPI HW-view and the AIS SW-View

### ***AIS Diagnostics Management Framework (no spec)***

- A service and framework that provides a single model, structure and view to ‘any’ (HW and/or SW) tests and diagnostic capabilities.

# Where HPI and AIS meet

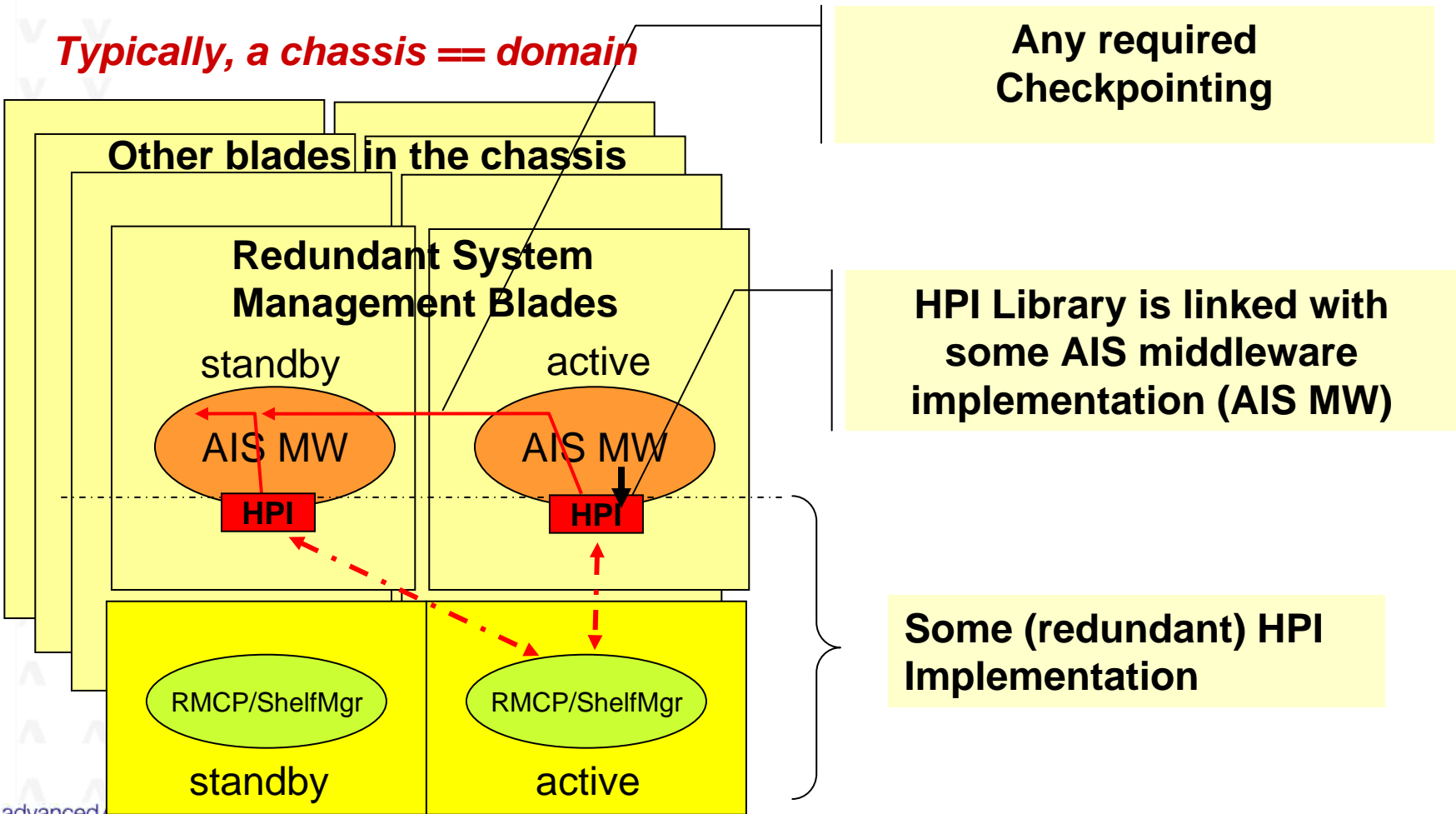
- HPI & AIS Working Groups collaborate so the layers cooperate



# Use case: HPI as service provider to Middleware

- Requirement: HPI Access must have no single point of failure

*Typically, a chassis == domain*

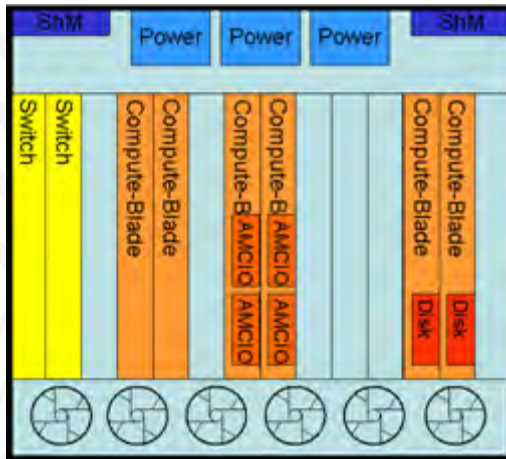


Shelf Managers

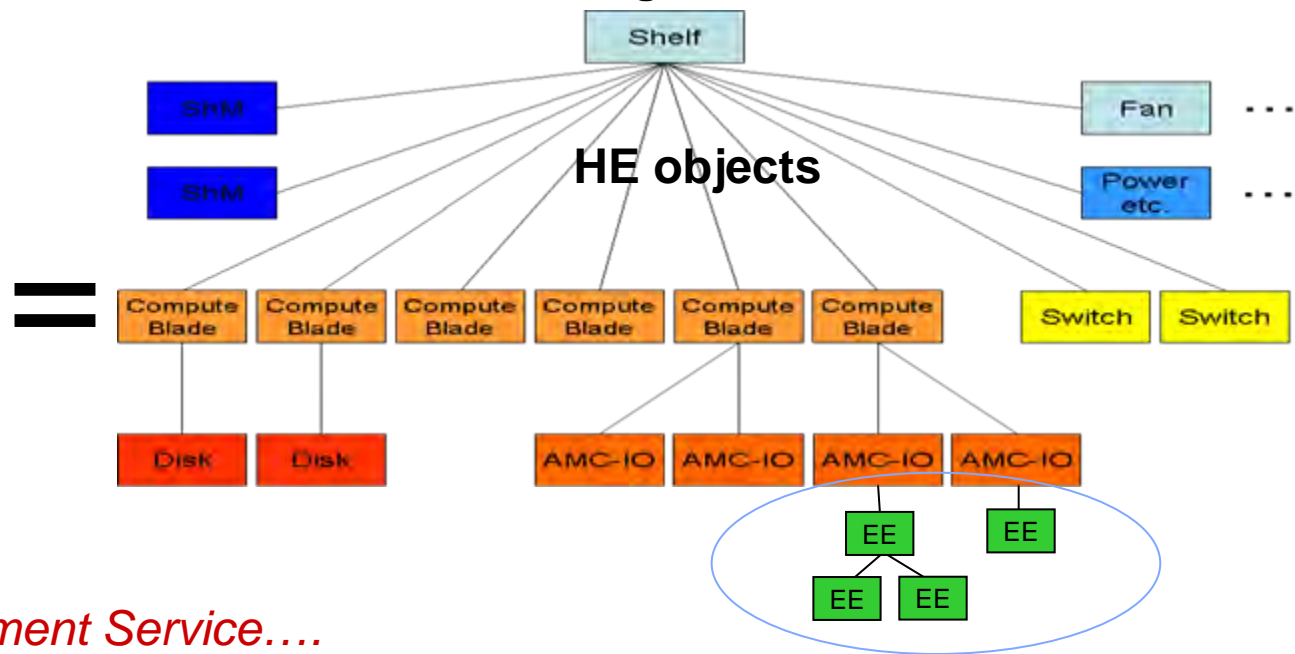
# Platform Management (PLM) Service

- IMM objects describe containment relationships between Hardware Elements (HE) and Execution Environments (EE)

## Physical



## Logical

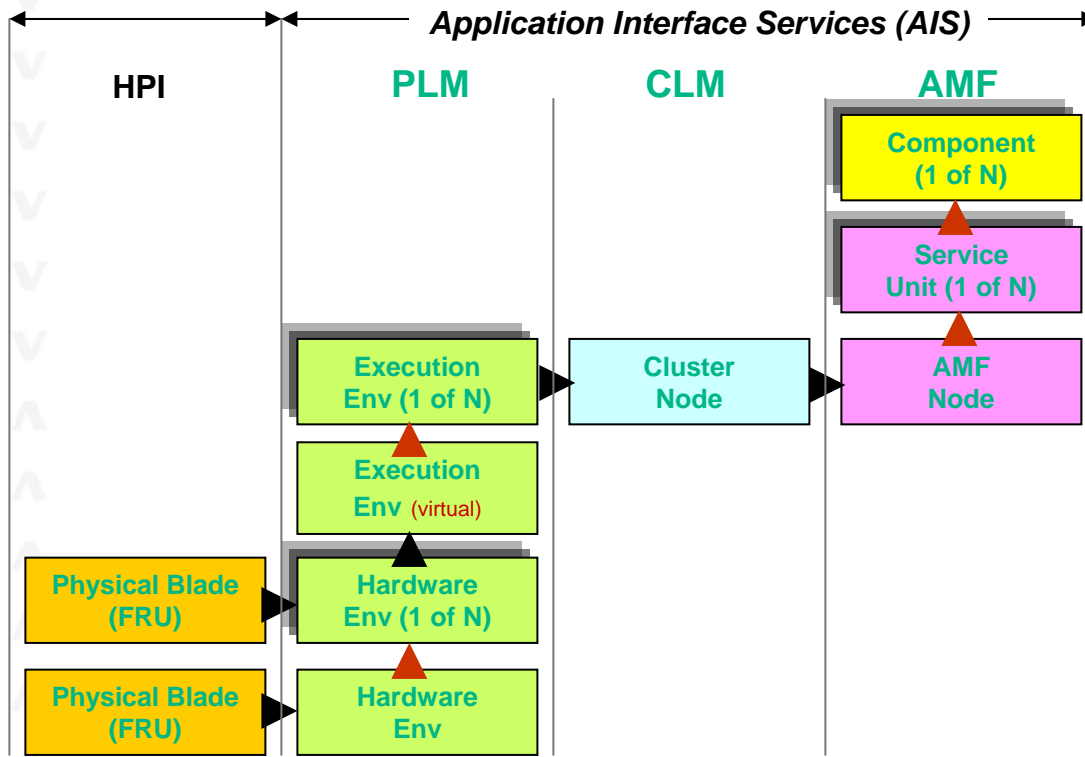


*Platform Management Service....*

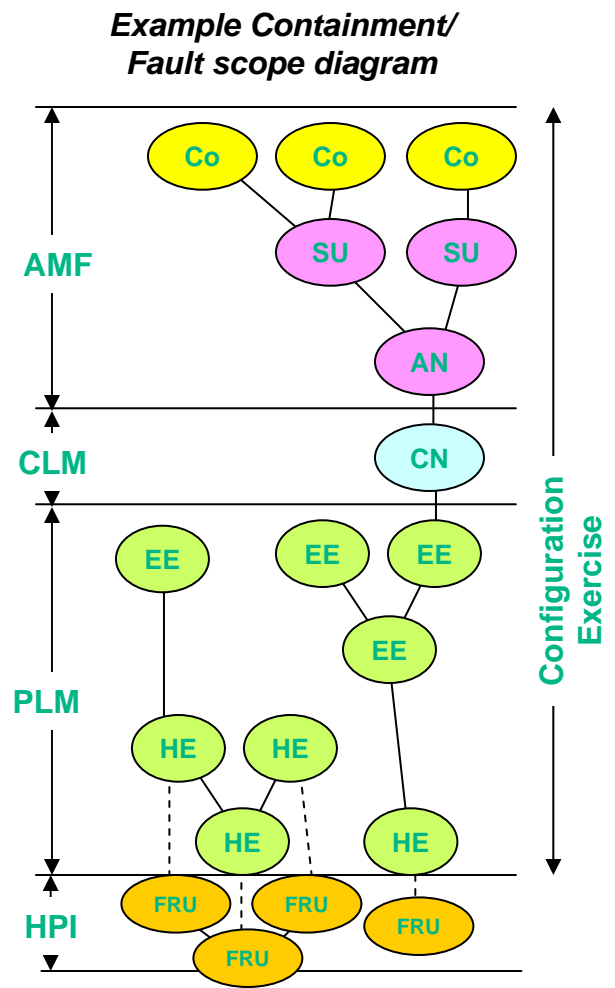
- *provides a bridge between HW-view and configured SW-view*
- *Connects the AIS information model with the discovered HW*
- *Provides administrative operations and configuration of hardware objects*

# Big Picture: Information Model

## Relationships and Dependencies



- ▶ - One to one relationship
- ▲ - One to many relationship
- - Your code/processes



Thank You

# BACKUP