



Education

Storage Management from SMI-S to Management Frameworks

Mark A Carlson

SNIA Technical Council, Sun Microsystems

SNIA Legal Notice

- The material contained in this tutorial is copyrighted by the SNIA.
- Member companies and individuals may use this material in presentations and literature under the following conditions:
 - ◆ Any slide or slides used must be reproduced without modification
 - ◆ The SNIA must be acknowledged as source of any material used in the body of any document containing material from these presentations.
- This presentation is a project of the SNIA Education Committee and the SNIA Technical Council.

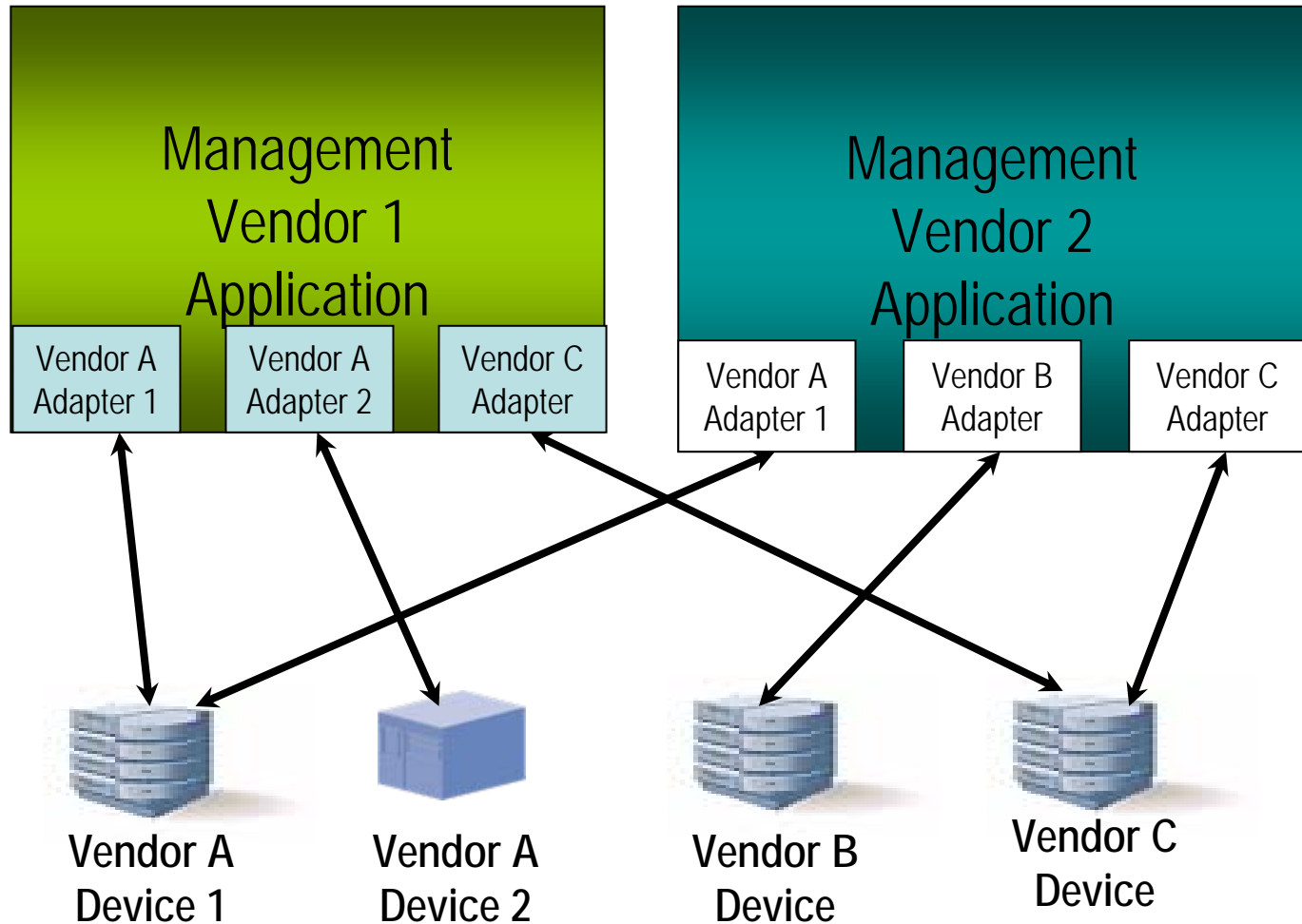
Topics

- What problem does SMI-S Solve?
- SMI-S is an Object Oriented API
- An Object Oriented Service API for Storage Arrays
- SMI-S moving into Shared Services
- Why SMI-S for DLM/ILM?
- SMI-S Automation of ILM using Services
- New Work: Management Frameworks

Problem: Management Standard for Agents

- Multiple vendor user interfaces
 - ◆ Discontinuity leads to inconsistency, errors
- Device support for applications
 - ◆ Each vendor must create adapters for proprietary APIs
- No opportunity for automation to reduce the cost and complexity

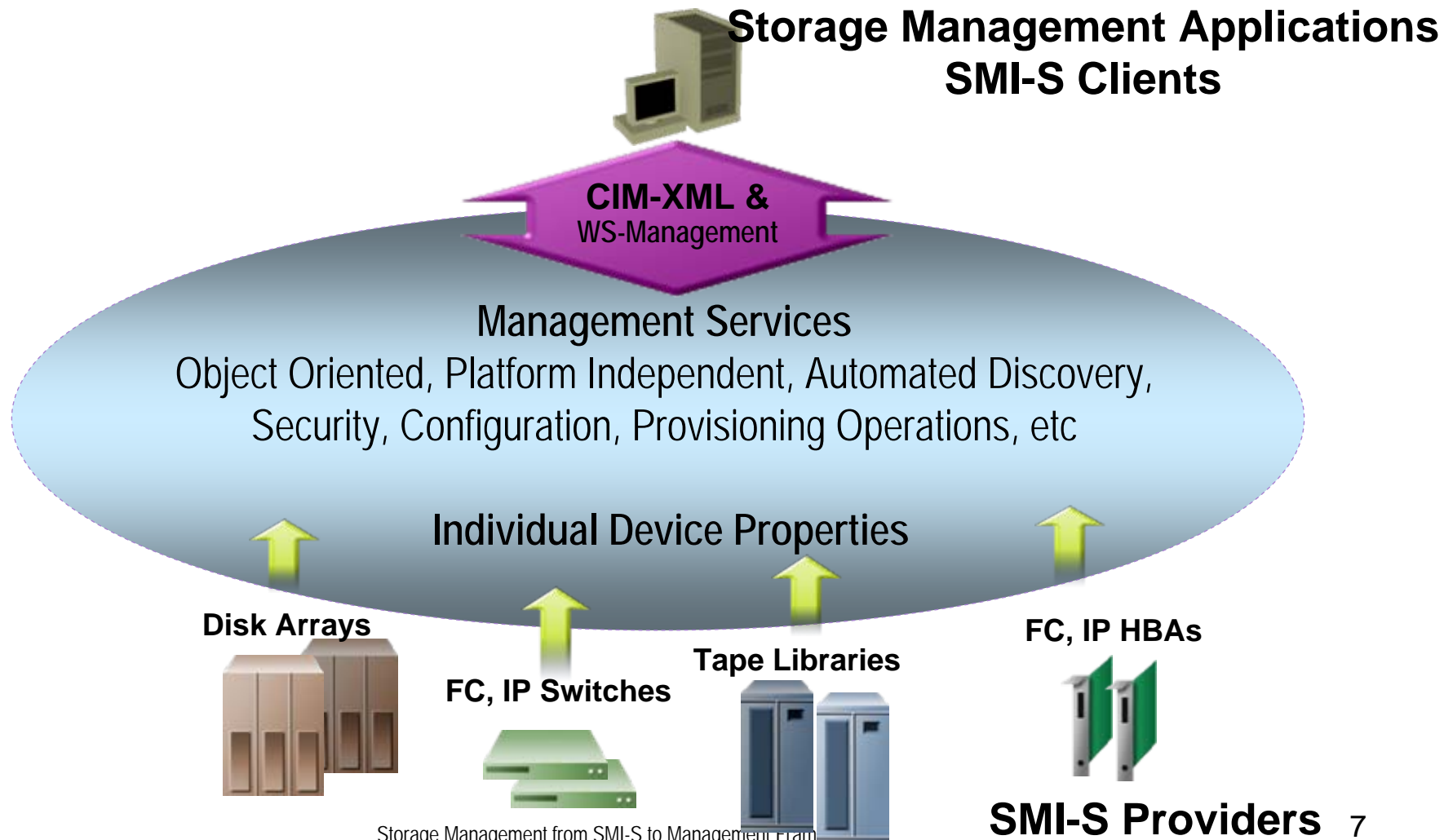
Multiple Proprietary APIs



from SMI-S ...*

*The Storage Management Initiative Specification (SMI-S) is an ISO and ANSI standard for interoperable management of storage resources

SMI-S: Services for Management



Object Oriented APIs

SMI-S is an object oriented API for managing storage devices

- Part of API is in common with other vendors
 - ◆ Base class(es) have common properties
- Rest of API is an extension of the Base Classes
 - ◆ Proprietary properties extend the common ones
- No need to swap APIs
- One transport (protocol) for all functions
 - ◆ Public and Private
- API exposed as a **Service**

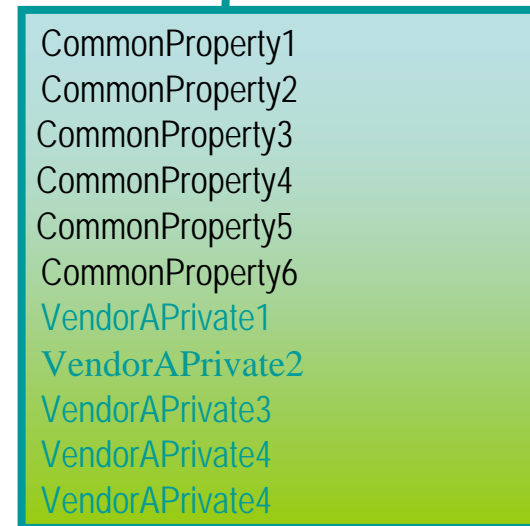
Object Oriented APIs

Each vendor adds their value to the standard in such a way as to enable clients to understand the common functions

Standard API Class(es)



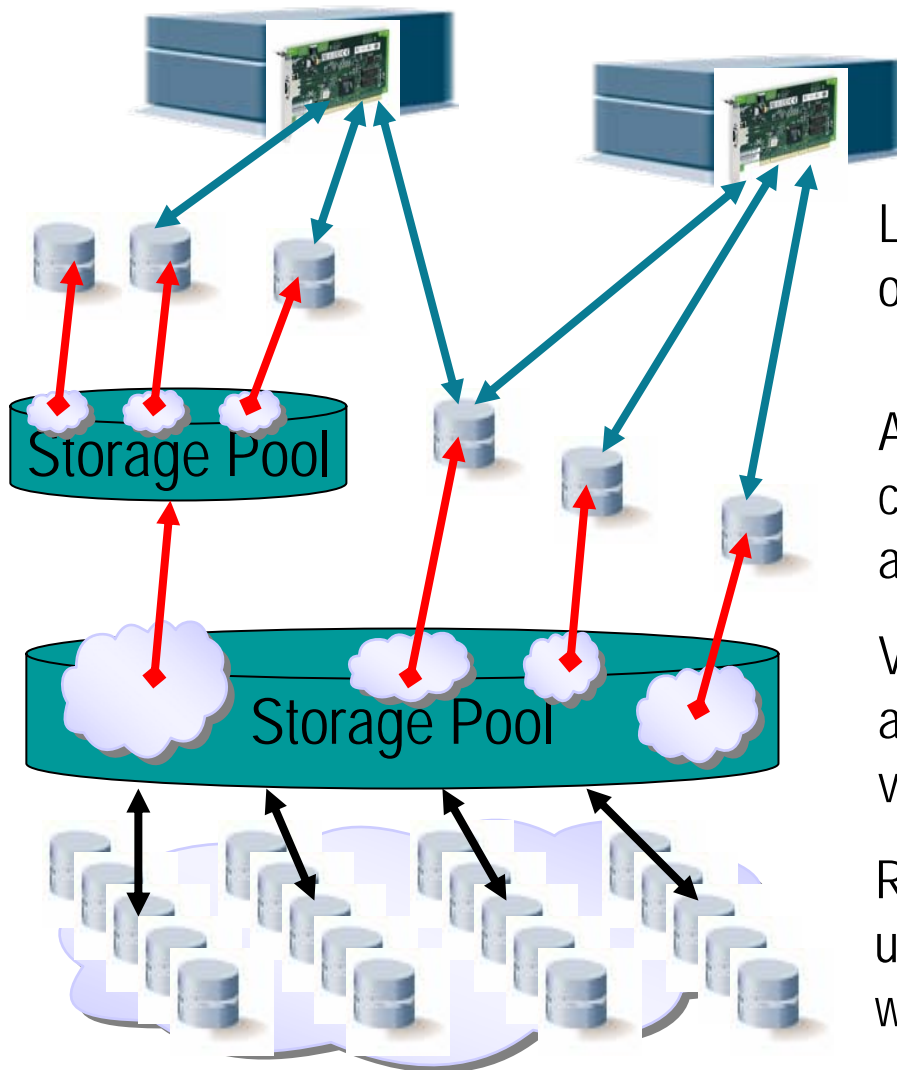
Vendor B's API



Vendor A's API

- SMI-S consists of multiple *Profiles*
 - ◆ Implementations can be thought of as storage *services* implementations (available on the network)
 - ◆ The profiles document the API to each type of device/resource
- SMI-S profiles specify how to manipulate the ***Information Model*** of the device/resource
 - ◆ Based on CIM, Accessed through multiple protocols
- The Array Profile in SMI-S forms an *Object Oriented Service API* to monitor and control Storage Arrays

Modeling an Array and its operations

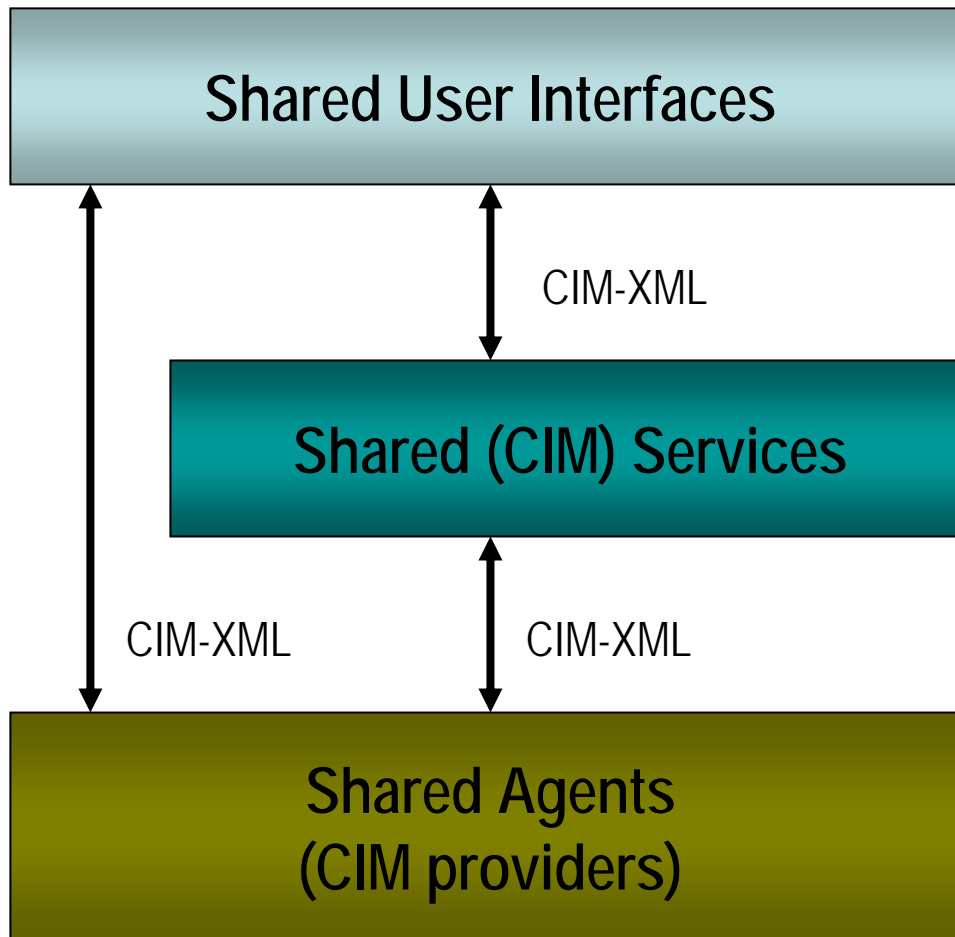


LUN Masking and Mapping is done to one or more host HBA fibre channel ports

A Setting is provided for the volume during creation which specifies performance and availability (QoS)

Virtualization of storage at the pool level allows either refined storage (pools) or virtual volumes to be created

Raw disks are aggregated into an undifferentiated pool of storage from which LUNs can be allocated



Presentation Tier

- GUIs, BUIs, CLIs

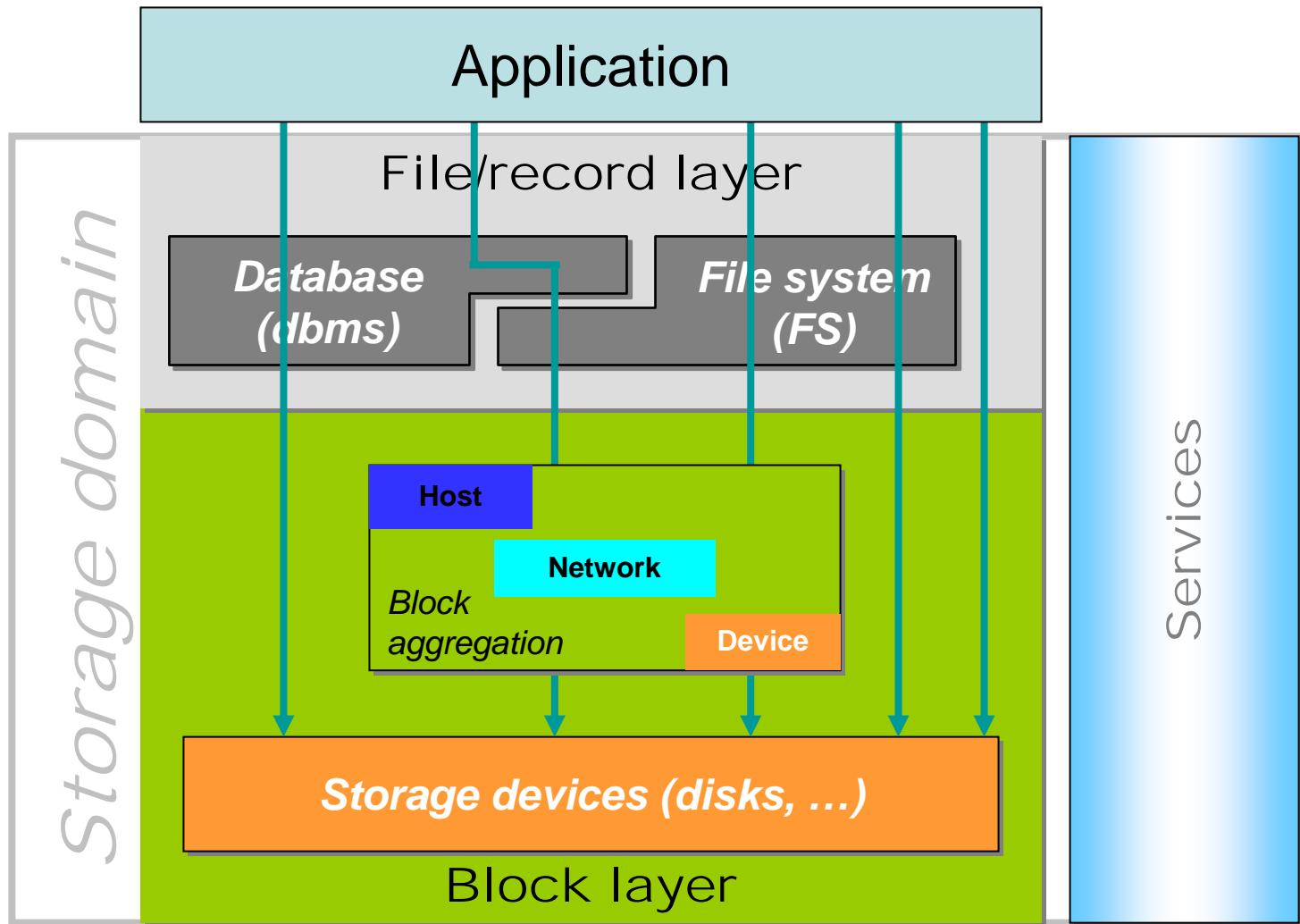
Services Tier

- Security, Policy
- Data Management
- Core Framework Services

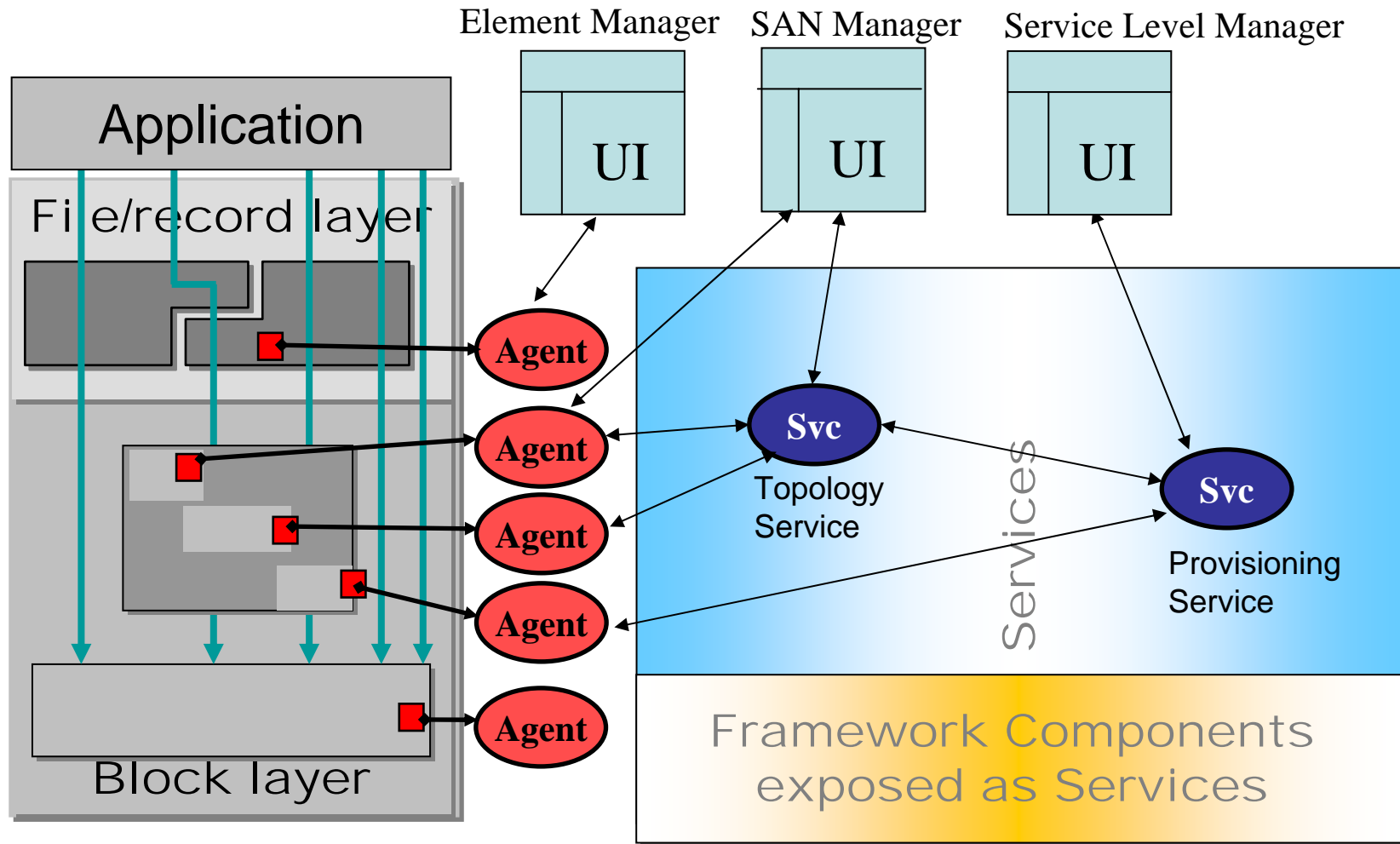
Agent (Device) Tier

- Device Instrumentation
- CIM Object Managers

The SNIA Shared Storage Model



Services for the Control Path



Foundation to build interoperable services for any storage resource 4

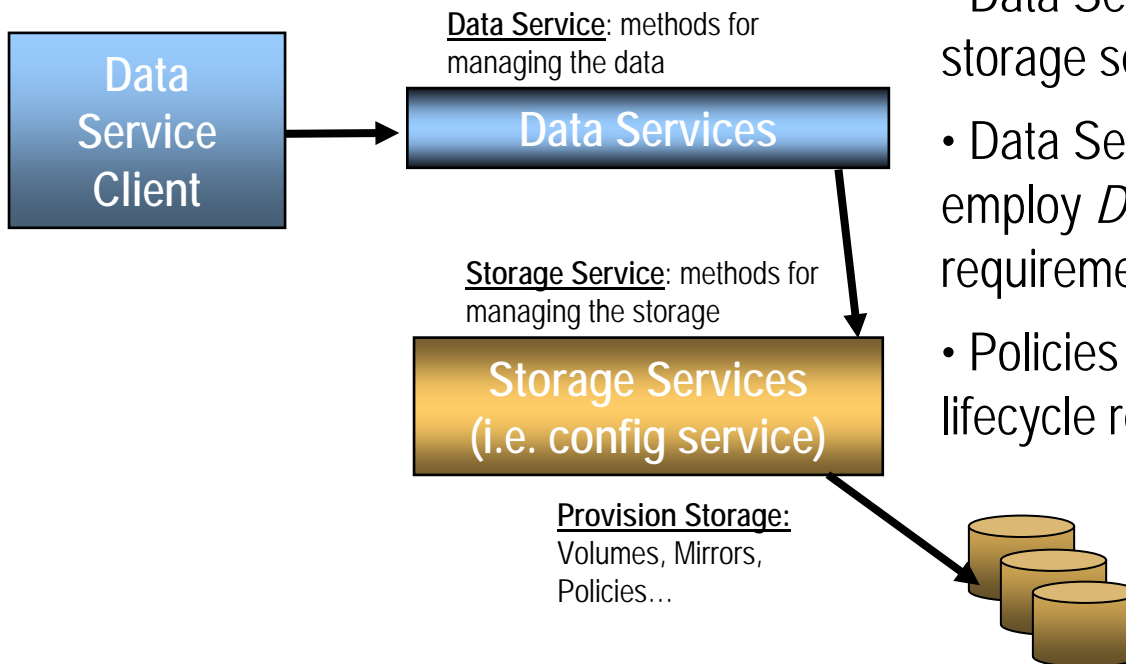
Domains for Management

- All of Management can be decomposed along two vectors
 - ◆ The type of resource that is being managed and the type of management done to those resources
- Resource Domain
 - ◆ The class of resource to which a particular device/resource belongs
 - ◆ Example: Storage Domain, Data Domain, Information Domain
 - ◆ Example: Server Domain, Network Domain, Application Domain
- Management Domain
 - ◆ The class of management to which a particular function belongs
 - ◆ Example: Fault, Configuration, Accounting, Performance and Security

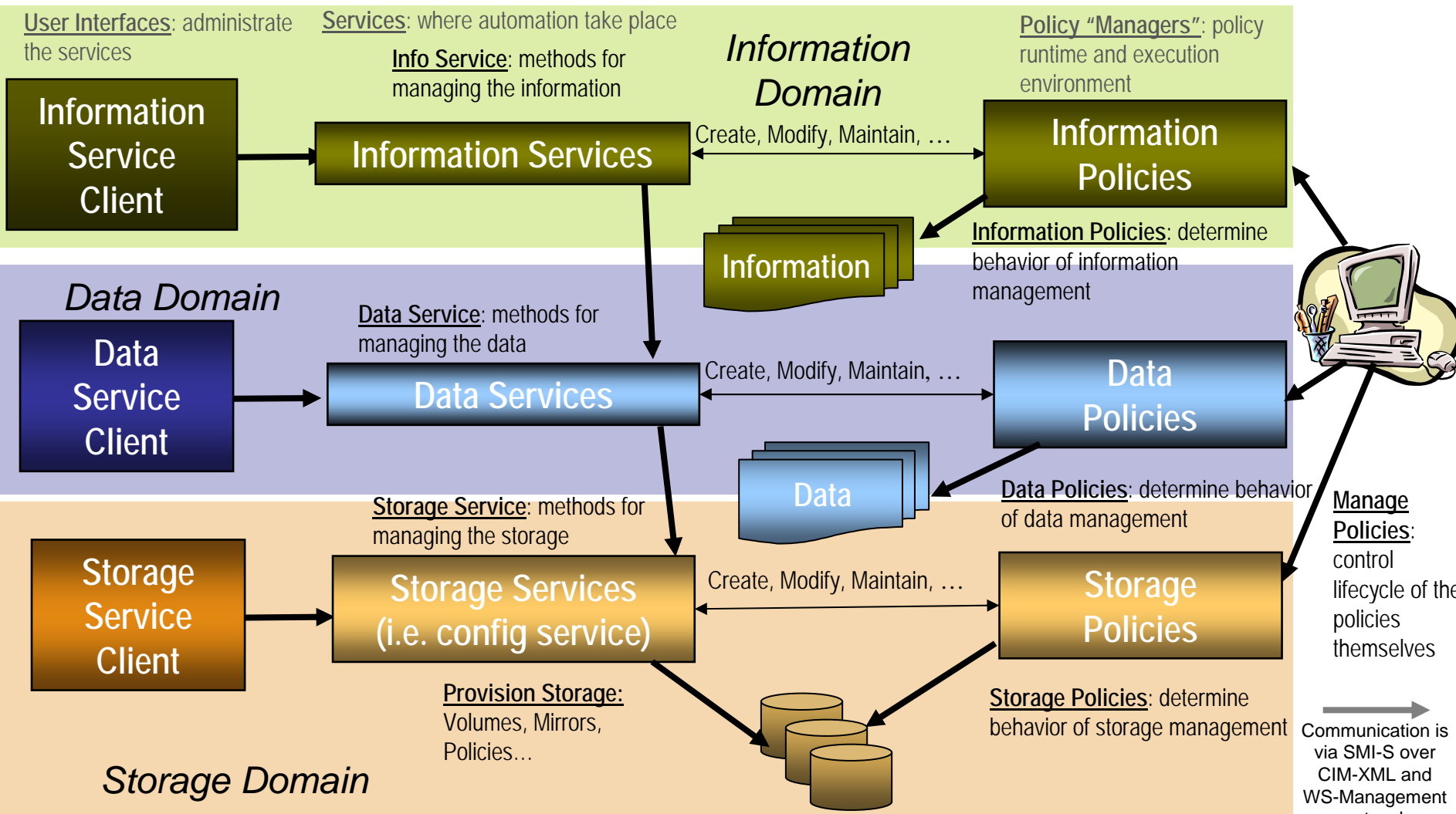
Data Services

Data Services are management services in the Data **Resource Domain** - managing the type of resource called "Data"

- Data Services shepherd the data
 - Maintain Copies
 - Play role in the Data's Lifecycle
- Data Services may be a *Client* of the storage services
- Data Services (or their policies) may also employ *Data Movers* to satisfy lifecycle requirements
- Policies can be employed to maintain lifecycle requirements

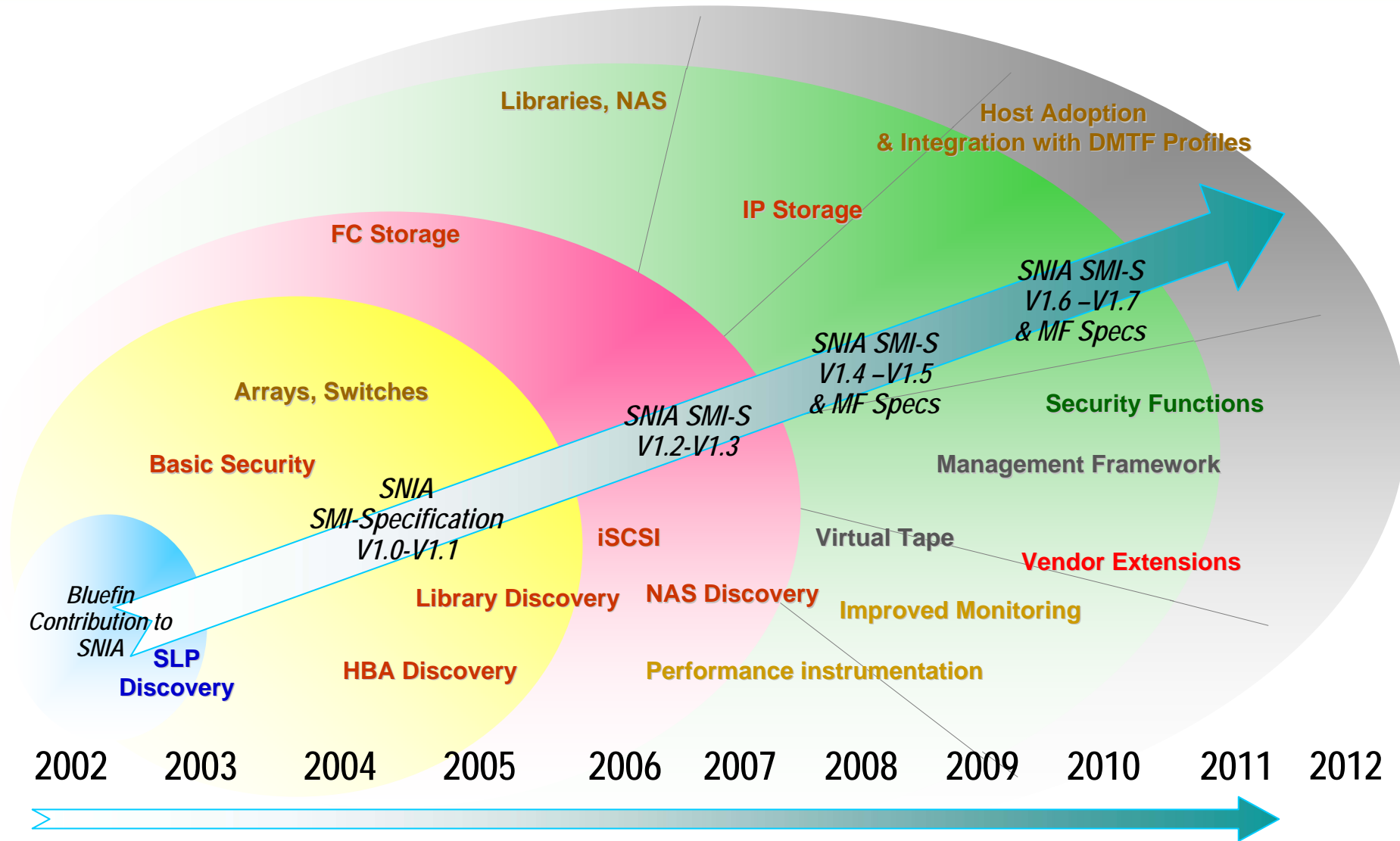


Composing Services in Resource Domains



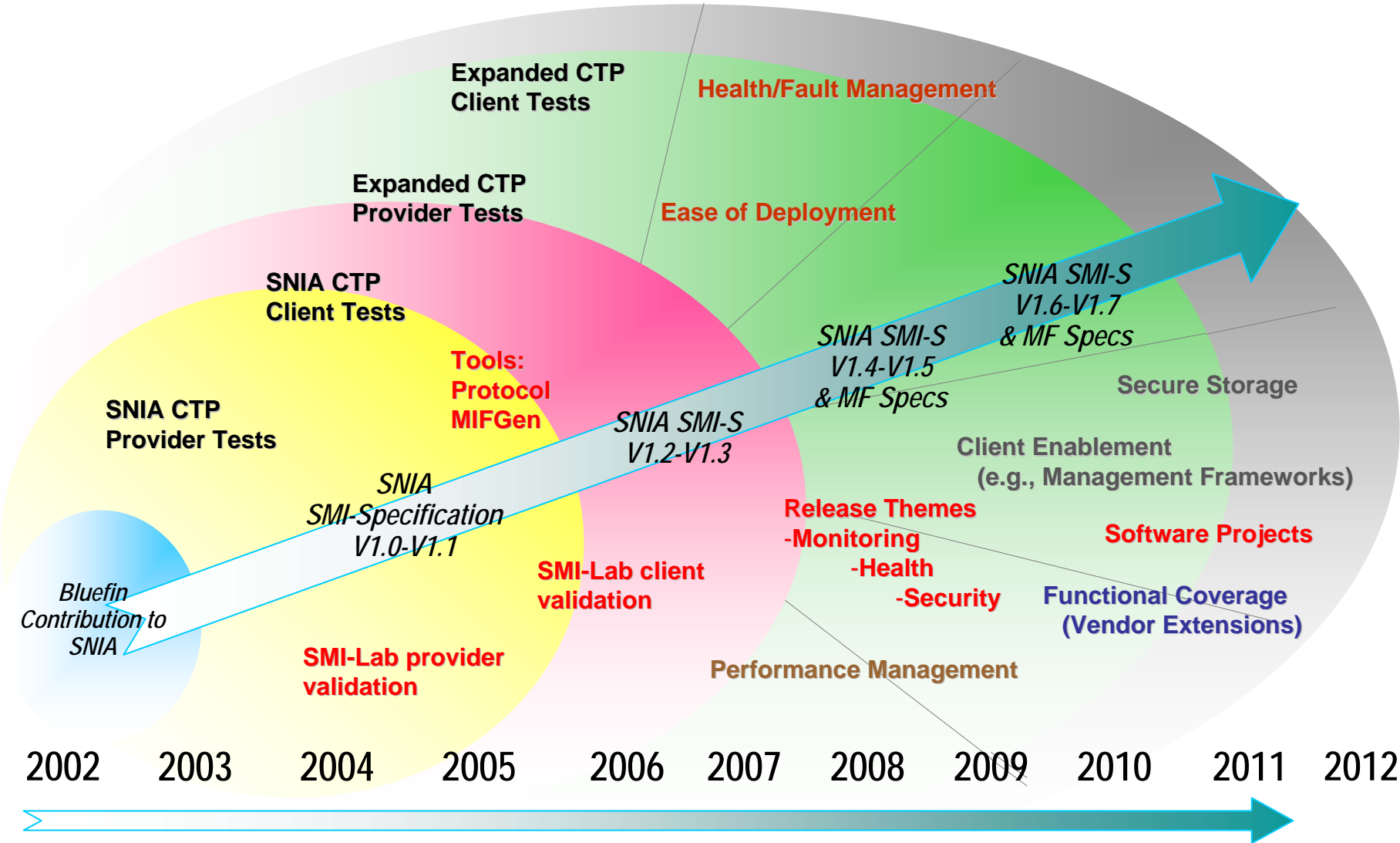
Storage Management Initiative

Technical Road Map – The journey continues



Storage Management Initiative

Ecosystem Road Map – The journey continues



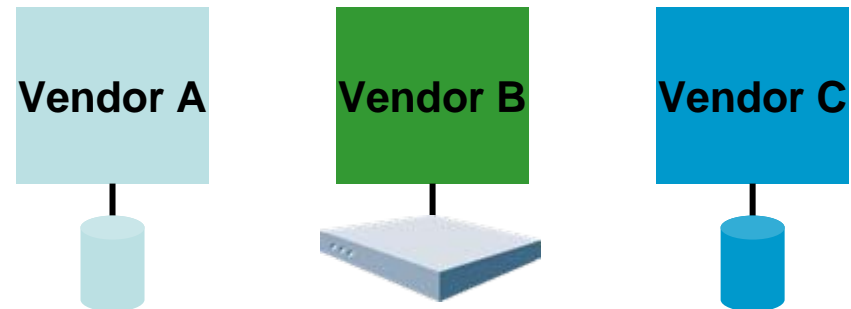
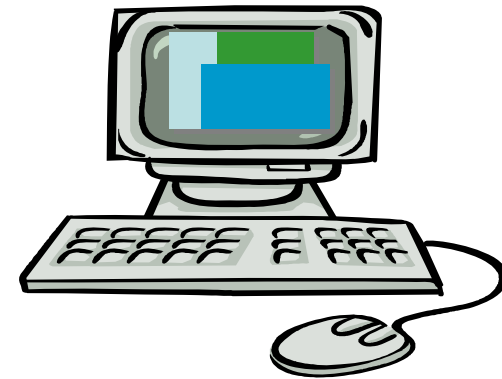
*... to Management Frameworks**

*A Management Framework is a set of common components, exposed as reusable services for use by management applications and other management services, components, etc. in the management environment.

- Phase 1
 - ◆ Individual Vendor Management Tools
- Phase 2
 - ◆ Enterprise Management Applications
- Phase 3
 - ◆ SMI-S Integration
- Today's issues and how to solve them
- Phase 4
 - ◆ Management Frameworks and Common Components

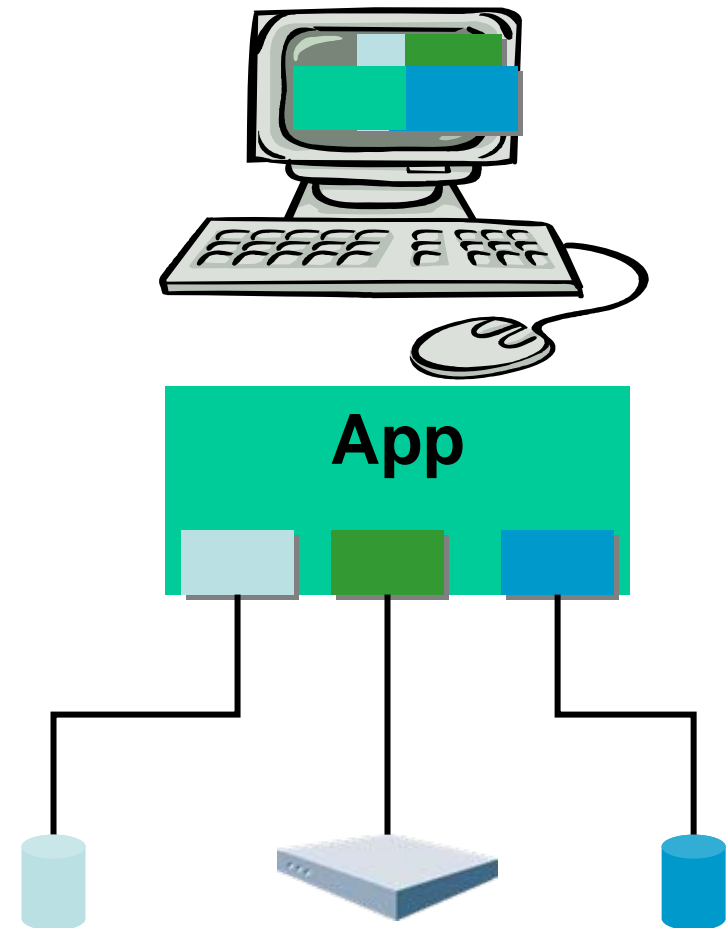
Phase 1 – Individual Vendor Management Tools

- Limited integration among tools
... maybe “link and launch”
- Vendor Differentiation:
 - ◆ Device by device functionality but new device ... new interface.
- Value
 - ◆ Support device administration
- End User Perspective:
 - ◆ High operational costs (training, platforms, integration)
 - ◆ Limited ability for user to deploy multivendor components
 - ◆ No commonality



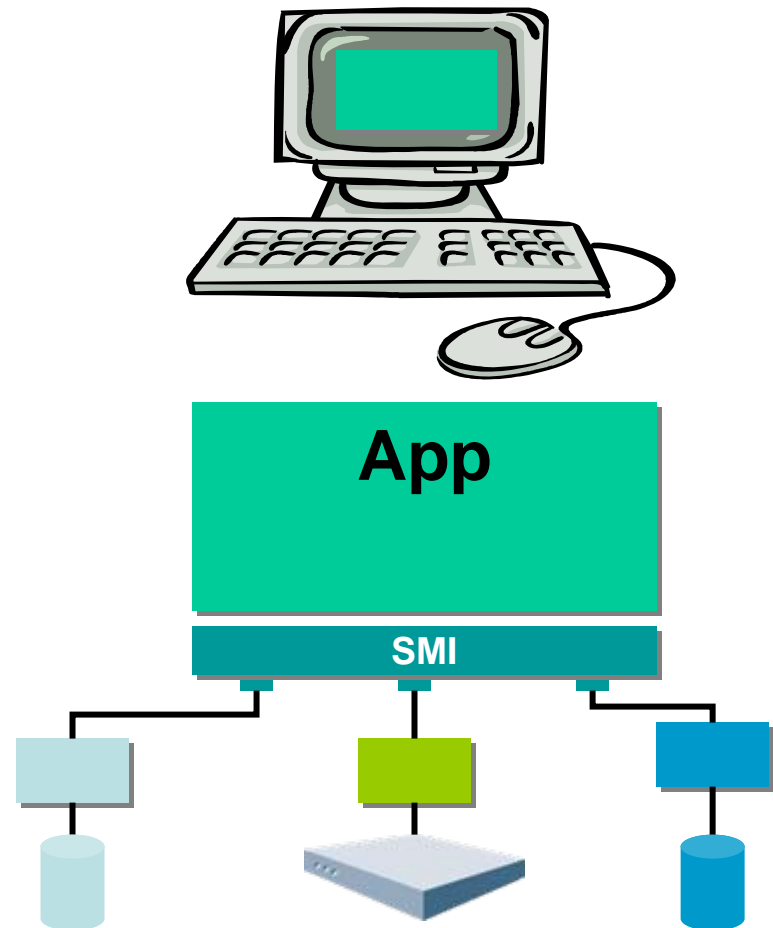
Phase 2 – Enterprise Management Applications

- Integration of device management on a single application
- Vendor Differentiation:
 - ◆ Breadth of device support
 - ◆ But ... expensive to maintain and improve
- Perceived Value
 - ◆ Broad device support
- End-User Perspective
 - ◆ Not enough device coverage
 - ◆ Limited to no improvement in functionality ... a box of tools



Phase 3 – SMI-S Integration

- Standard interface to devices
- Vendor Differentiation:
 - ◆ Increased support of devices
 - ◆ Using a “standard” interface
- Value
 - ◆ Device support -> automation, policy
- End-user perspective
 - ◆ Broader access to devices
 - ◆ Common management “profiles”
... similar operations across dissimilar devices

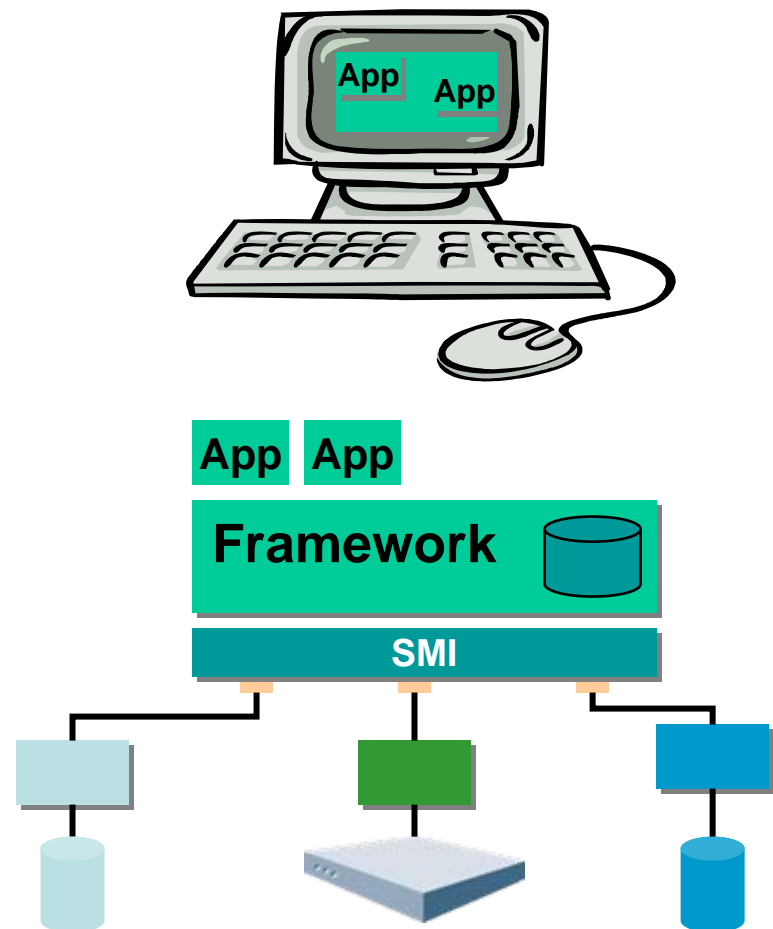


One of the problems

- Time to market
- Vendor develops new functionality
 - ◆ Standard takes a year
 - ◆ Provider Implementation another 6 months
 - ◆ Client implementation from 2 Years to Never
- Need to accelerate delivery to end user faster

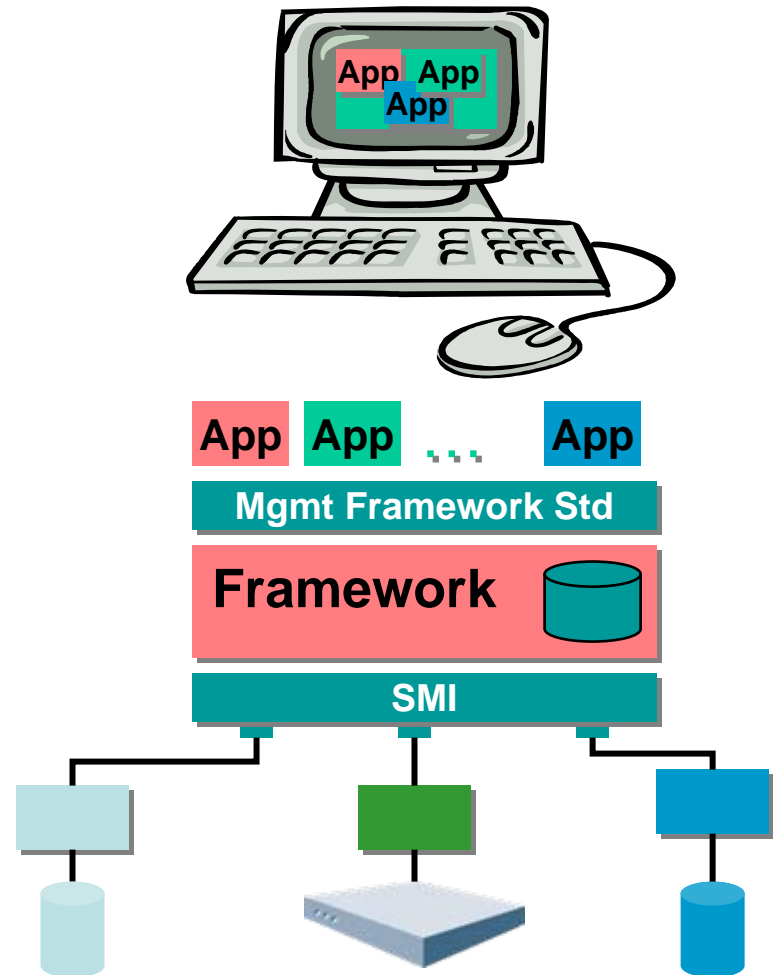
Phase 4 – Framework and Modular Applications

- Integrated framework and repository for end-to-end device view
- Vendor Differentiation:
 - ◆ Deliver functionality across multiple devices, e.g. storage provisioning
 - ◆ Modular applications with scaled functionality
 - ◆ Integrated information model
- Value
 - ◆ Framework -> Applications
- End-user perspective
 - ◆ Emergence of integrated management
 - ◆ But ... single vendor lock-in
 - ◆ Storage management still did little to reduce operational costs.



The new opportunity

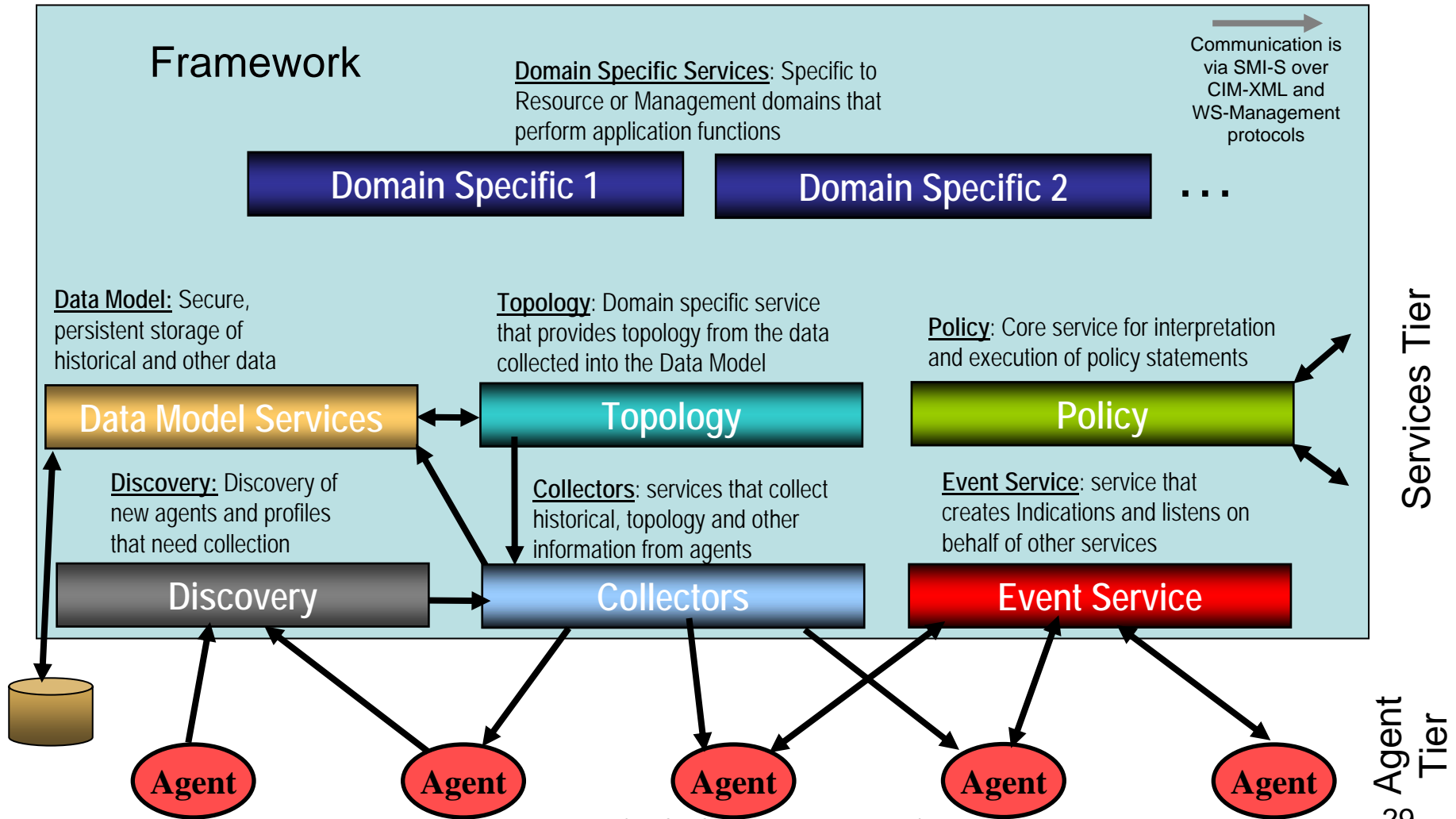
- Standard framework interface(s) for use by storage management applications
- Support for multi-vendor applications using the common framework
- Vendor differentiation
 - ◆ Leverage framework investment across multiple vendors
 - ◆ Ability to support applications from multiple partners
 - ◆ Easier to integrate support from SMI devices
- Benefit to users:
 - ◆ Access to a broader set of functionality from multiple vendors
 - ◆ Results in lower cost of management



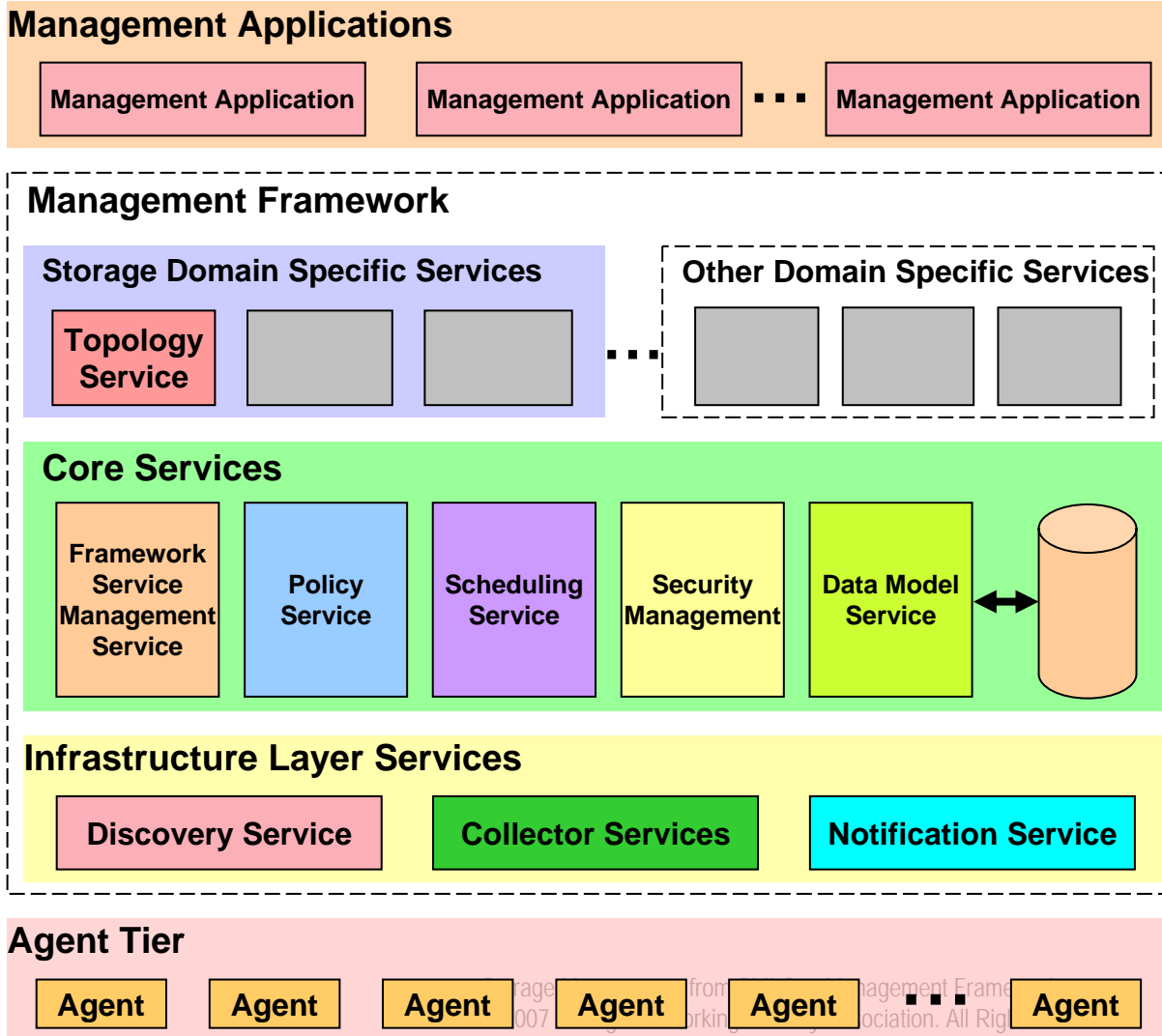
Management Frameworks

- In order to support the shared services, a set of common components, factored out of management applications, should be standardized
 - ◆ Similar to the CIM Object Manager for Agent Tier
- This work will accelerate SMI-S Client development
 - ◆ Leverage standard frameworks for applications
 - ◆ Customers get to choose best of breed management functions from multiple vendors
- Services are available both locally and remotely

Anatomy of a Framework



High Level Architecture



The layers of the architecture include:

- Infrastructure Layer Services
- Core Services
- Domain Specific Service

- **Discovery Service**
 - ◆ The Discovery Service uses standard mechanisms to discover new Agents and supported SMI-S profiles/sub-profiles.
- **Collector Service**
 - ◆ Collector Service gathers information from agents in a defined domain and feeds the information to the Data Model Service
- **Notification Service**
 - ◆ Centralized event collection, normalization and distribution service

Discovery Service

- The Discovery Service uses standard mechanisms to discover new Agents and supported SMI-S profiles/sub-profiles....
- Discovers agents, profiles, sub-profiles and implementation options.
- Provides a well-defined means, via the Data Model Service, to...
 - ◆ Specify the discovery granularity within a profile
 - ◆ Communicate this discovery scope, depth, and breadth, to Collectors.
- Provides a means to specify 'implementation options' for discovery:
 - ◆ Optional subprofiles
 - ◆ Optional elements within a profile
 - ◆ Vendor extensions
- Uses SLP as required to discover SMI-S instrumentation.
- May be extended in the future by using additional discovery mechanisms.

Discovery Service – continued

- Provides a standard way to notify other services about discovered resources (subscribe-notify) via the Notification Service.
 - ◆ Note that services may also learn of discovered resources via the Data Model Service.
- Provides a standard way to configure the scope of discovery and discovery specific parameters.
- Supports the adding of new profiles to be discovered.
- Supports discovery of vendor defined extensions to a profile.

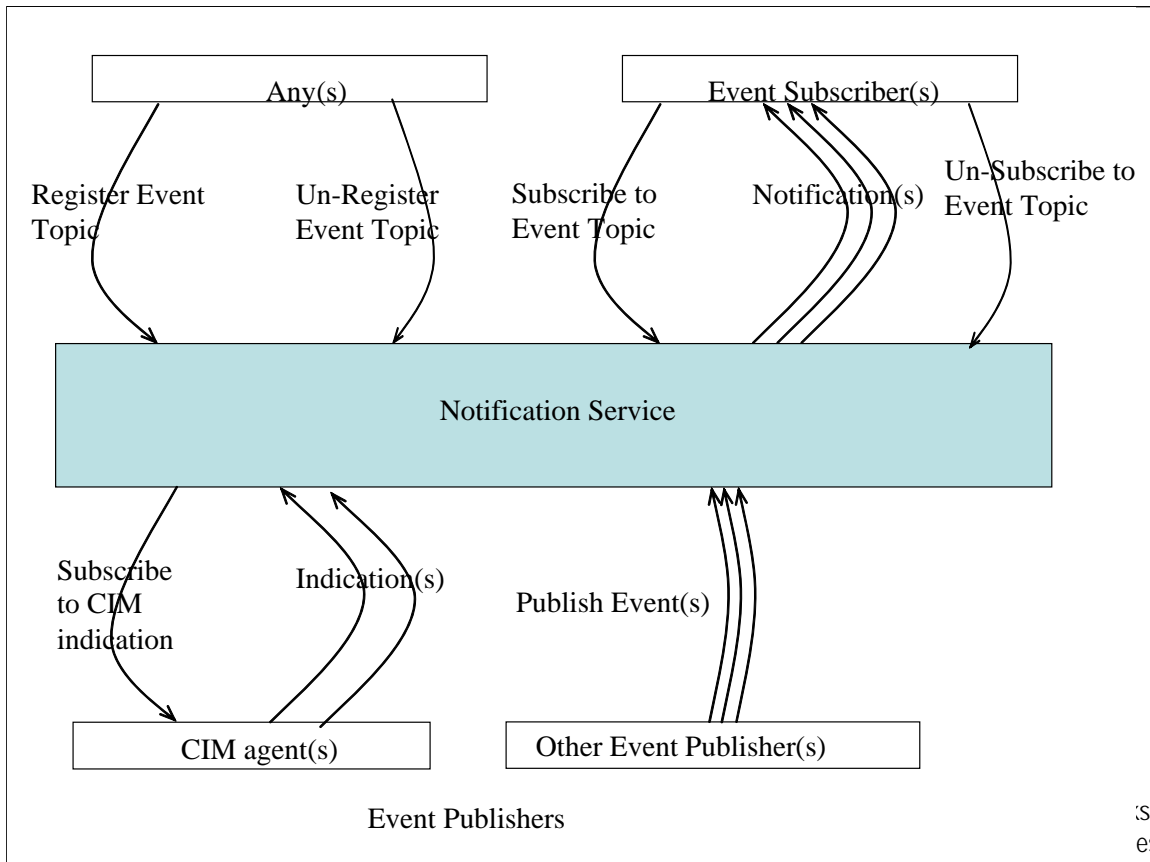
Collector Service

- Collector Services collect information from agents in a defined domain
- Collector Services models instantiated in the Data Model Service that reflect the collected information
- Collector Services can be specialized to a domain specific data model subset
- Multiple Collector Services coexist and may be defined by the type of information that is presented by the devices or installed software
- Collector Services are not expected to do any aggregation of information
- A single Collector Service may collect information from multiple types of devices or installed software
- A Collector Service shall provide granular control over what information is collected
- Collector Services shall declare events via the Notification Service

Collector Service - continued

- Collector Services need a management interface through which their capabilities can be made known
- Collector Services participate in security concerns with regard to accessing device or installed software management information
 - ◆ The Discovery Service needs to have already dealt with issues like device credentials to have done its job
 - ◆ The management application establishes what the credentials are
- Collector Services may deal with control for devices or installed software
 - ◆ This may only be for management framework elements, and would only apply where the model in the Data Model Service supports control operations
- Collector Services must report error situations to the Framework Service Management Service

Notification Service

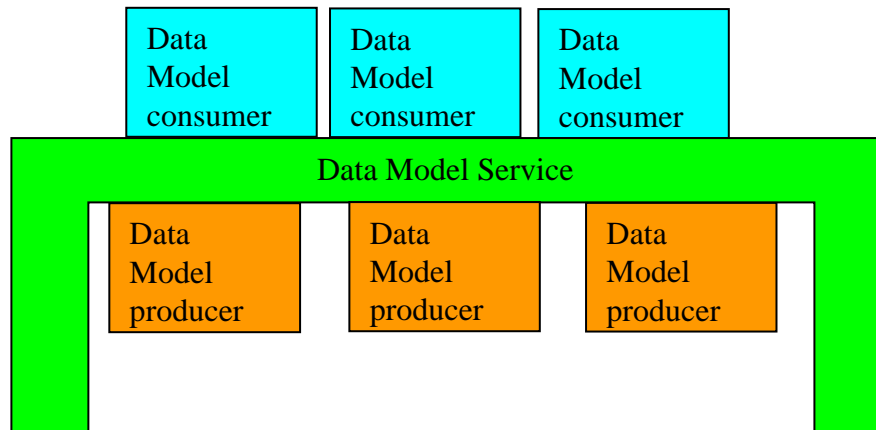


The Notification Service is a central clearing house for all asynchronous notifications in the system, either coming from the agent tier, or from other services

Publishing and subscribing is on the basis of *Topics*

Core Services

- **Data Model Service**
 - ◆ A common place to coordinate model based operations
- **Framework Service Management**
 - ◆ Provides a means of monitoring and controlling how the framework and its services use the execution environment.
- **Scheduling Service**
 - ◆ Provides the ability to initiate actions in management framework services on a persisted schedule.
- **Policy Service**
 - ◆ A service that executes policies against the rest of the framework



Any framework or other service can be both a producer and a consumer

Default producers can perform functions such as historical archive and initiation of operations against agents

The Data Model Service is similar to a CIMOM as it is a central place to get all instances of the information model (including historical ones)

Data Model producers (similar to providers) create model instances for consumption by other services

Data Model consumers (similar to clients) utilize and manipulate the instances

Data Model Persistence

- ▶ Intent is not to have a general purpose “Persistence Service” for use by services
 - ◆ They can always open up a database connection on their own
 - ◆ Provide persistence only in terms of the model
- ▶ Default Instance Provider
 - ◆ Persistent instances for use by services
- ▶ Default History Provider
 - ◆ Historical instances time stamped and managed

Default Instance Provider

- Similar to core part of CIMOMs
- Persisted instances to backing store (Database)
- Can be used by services to store configurations
- Can be used by services to store results of their operations
- Survives reboot
- Schema is dynamically provided (based on MOF or WS-CIM format)

Default History Provider

- Similar to Default Instance Provider
- Timestamps instances automatically
- History is managed by what consumers are interested in
 - ◆ Old instances are expired if no services want to consume them
 - ◆ Consumers configure the schema
 - How often are instances collected (by collectors)
 - How long are instances retained

Default Actuator

- Services wishing to manipulate the agents through the data model can call methods that are then propagated to the agents
- Allows for future transactional model
- Can be a common place to iterate over multiple agents
 - ◆ Handling partial failure
- Basis for Policy Actions

- The Policy Service is a core service providing an execution environment for expressions of policy.
 - ◆ Its main function is to act as a supporting service for these policies such that they each do not need to be their own service in the system.
 - ◆ Simple automation tasks can be added to the system without the full requirements of being a service.
 - ◆ Logic for a management application can then be a combination of policies and service logic composed as needed to perform the required functions.
 - ◆ Support for specific expressions of policy (CIM-SPL for example) will be mentioned by reference in the eventual interface specification.

- The Topology Service is a storage domain specific service that provides topology from the data collected into the Data Model.
- The scope of the Topology Service is all physical and logical storage network topologies, including but not limited to:
 - ◆ Fibre Channel (FCP, SB), iSCSI, SAS
- Creates topology data models in the Data Model Service as needed, based on requests from the other services. This includes:
 - ◆ Provide Data Path as an object for consumption by other services.
 - ◆ Provide data models that represent hierarchical aggregate groups of devices.
- Supports recipes for tracing end-to-end physical and logical paths.
- Provides status rollup (aggregate status) for data models representing groups of devices.
- Monitors the Data Model via the Notification service for changes in raw topology data, and updates topology data models in the Data Model Service as necessary.

- ▶ Please send any questions or comments on this presentation to SNIA:

trackstoragemgmt@snia.org Marty Foltyn/John Kelly

**Many thanks to the following individuals
for their contributions to this tutorial.**

SNIA Education Committee

**Management Frameworks TWG
Storage Management Initiative Marketing Committee
Ray Dunn
Edgar St. Pierre
John Crandal**



Education

Thank You!

Mark A Carlson, SNIA Technical Council, Sun Microsystems