

An Early Look at SNIA's Management Frameworks effort

Mark A. Carlson
Sun Microsystems, Inc.

With help from the Members of the SNIA
Management Frameworks Technical Working
Group

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 Management Frameworks TWG and the SNIA Technical Council.
- Feedback is limited due to SNIA IP Policy and the requirements on participation in a TWG

Overview

- What problem are we trying to Solve?
- Management Frameworks Overview
- Service Interface Decomposition
- High Level Architecture
- Common Aspects of Services
 - Security
- Infrastructure Services
 - Discovery Service
 - Collector Services
 - Notification Service
- Core Services
 - Data Model Service
 - Framework Service Management
 - Scheduling
 - Policy
- Domain Specific Services
 - Topology Service

Problem:

Management Standard for Clients

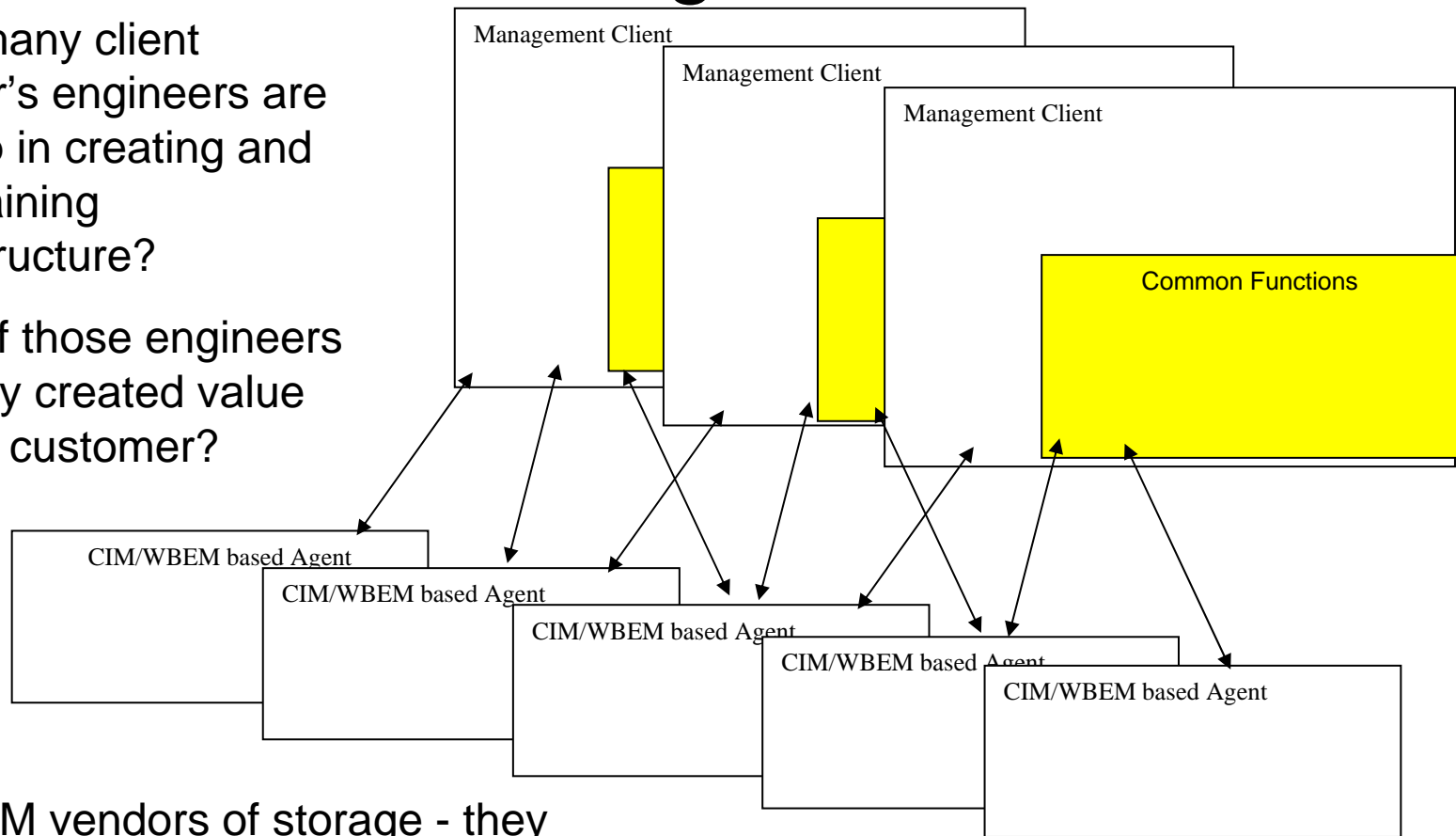
- CIM/WBEM (& SMI-S) have laid a good foundation in the agent tier, but...
- Each Management Client still:
 - Must create a large amount of infrastructure code before they can add value
 - Creates this from scratch each time
- Client functions cannot interoperate with and build upon other client functions
- Customers still get locked into a client vendor and are limited to that vendor's value
- This set of problems is independent of domains (Storage, Servers, etc.) and needs a domain independent solution



Re-inventing the wheel

How many client vendor's engineers are tied up in creating and maintaining infrastructure?

What if those engineers actually created value for the customer?



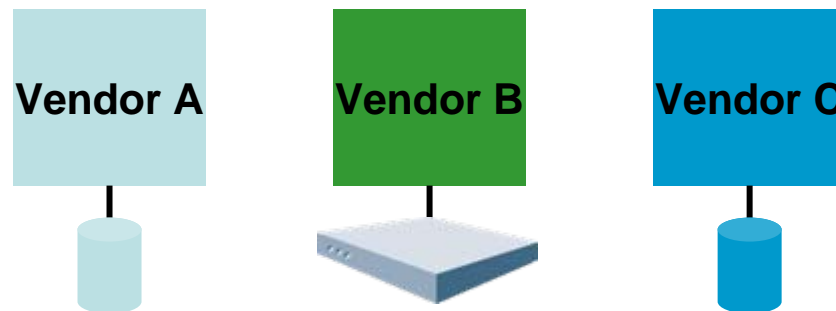
For OEM vendors of storage - they need to provide their value independent of any particular client

The Evolution of Management Solutions

- 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

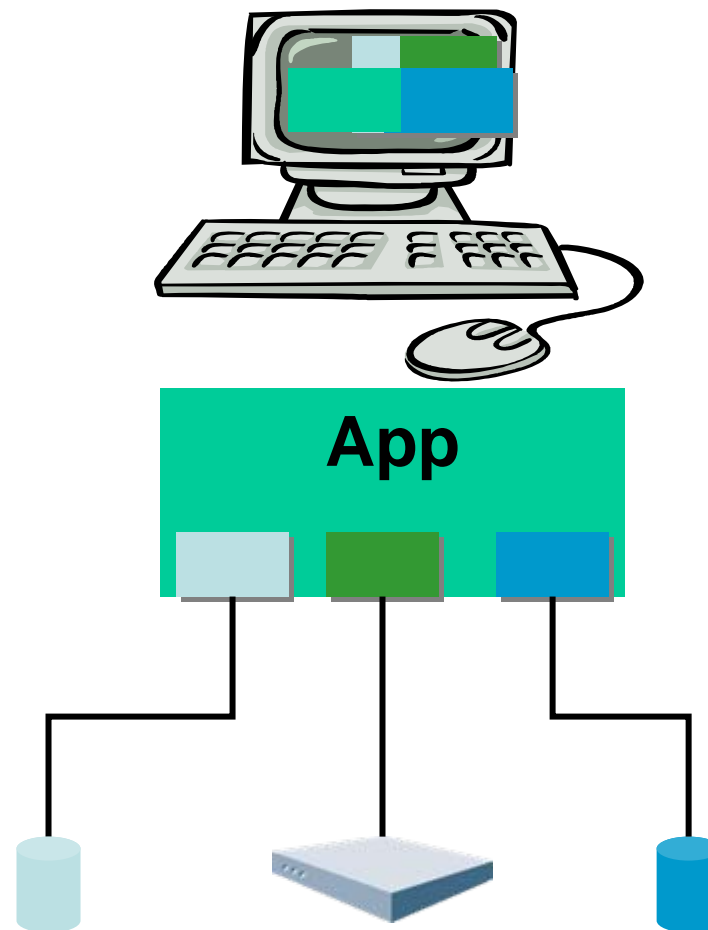
Phase 1 – Individual Vendor Management Tools

- 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



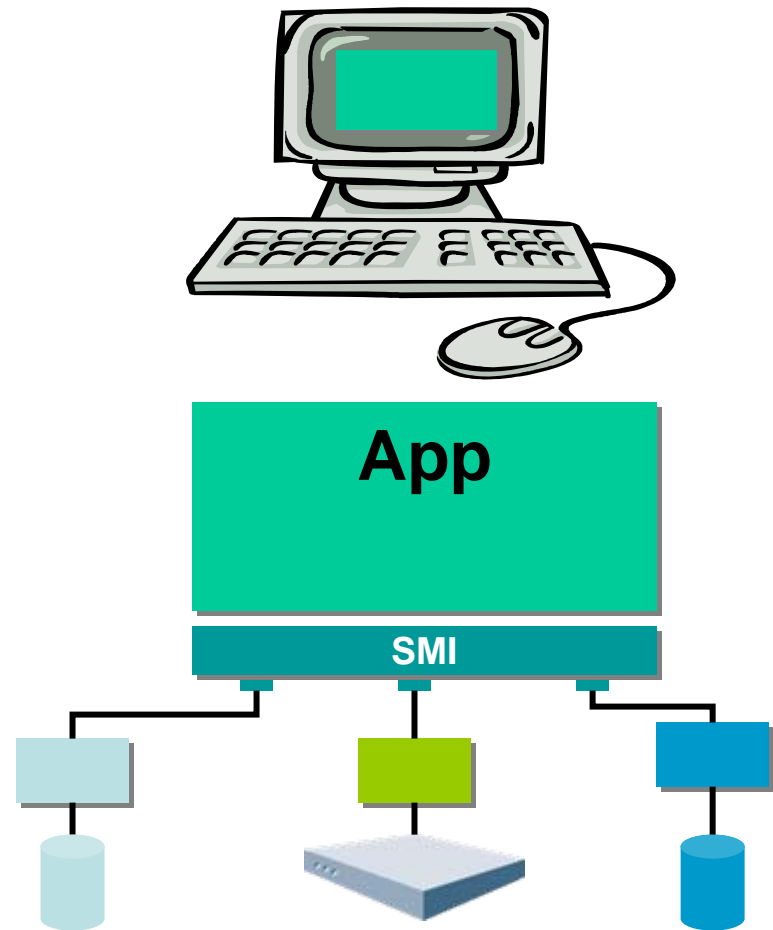
Phase 2 – Enterprise Management Applications

- 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

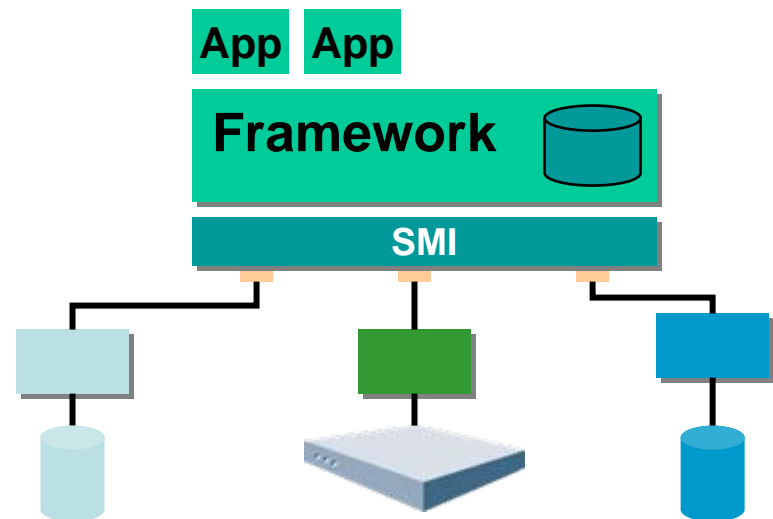


One of the problems

- 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

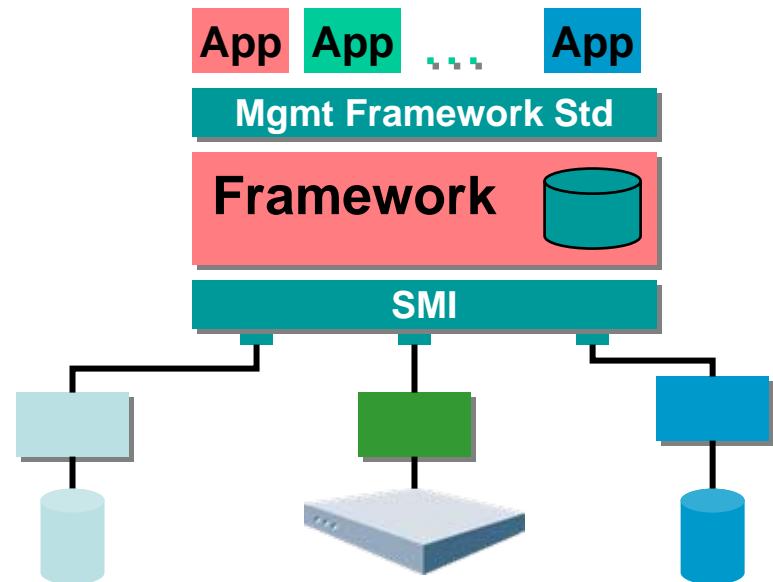
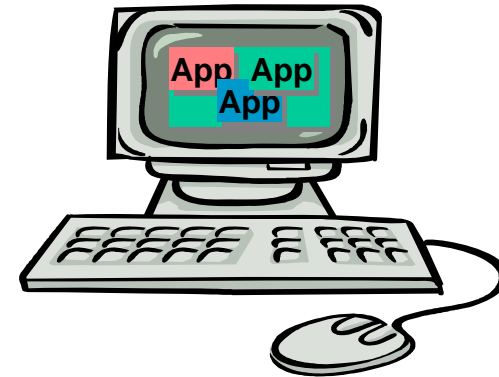
Phase 4 – Framework and Modular Applications

- 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.

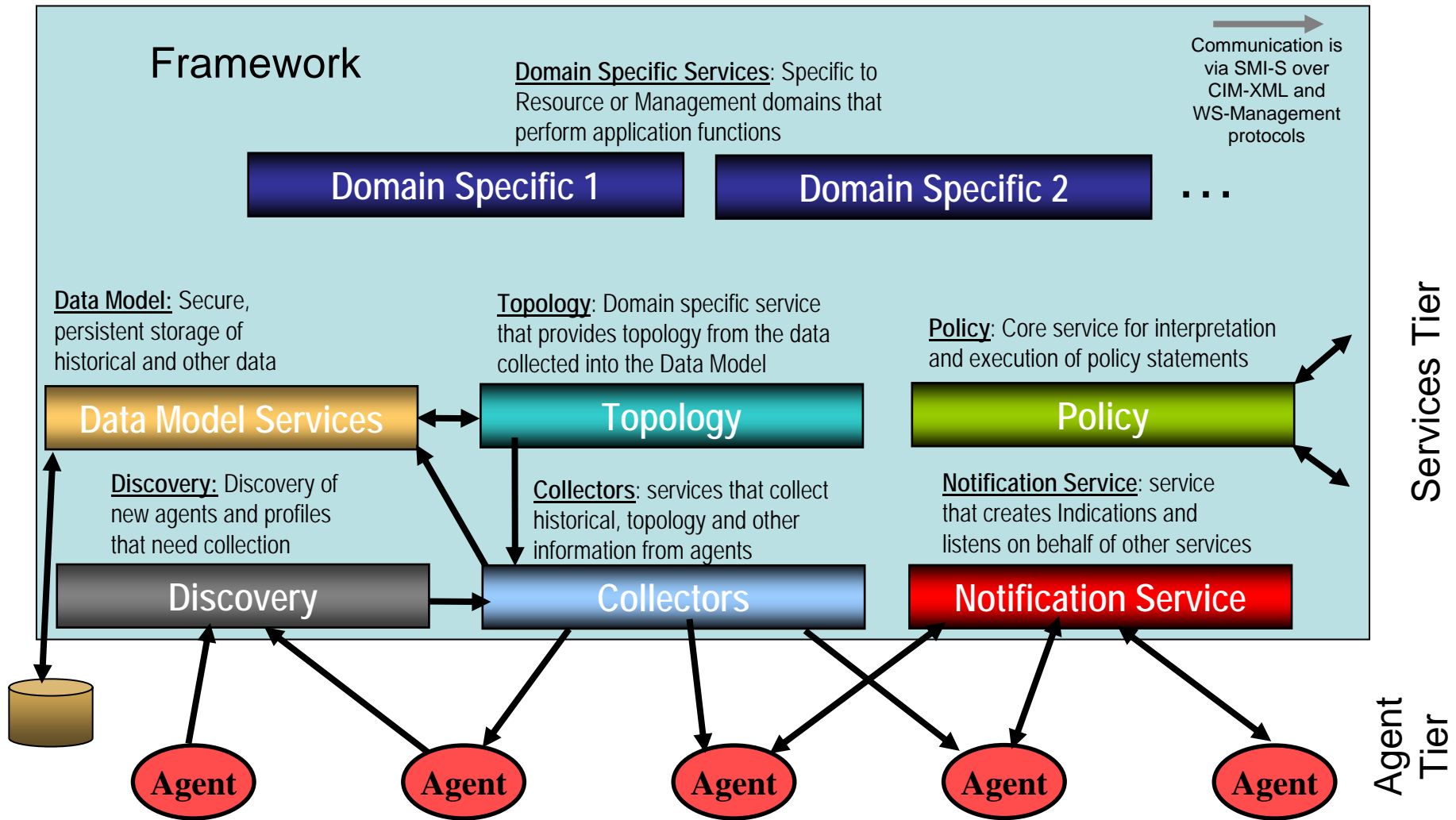


The new opportunity

- 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 supports 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



Framework Overview

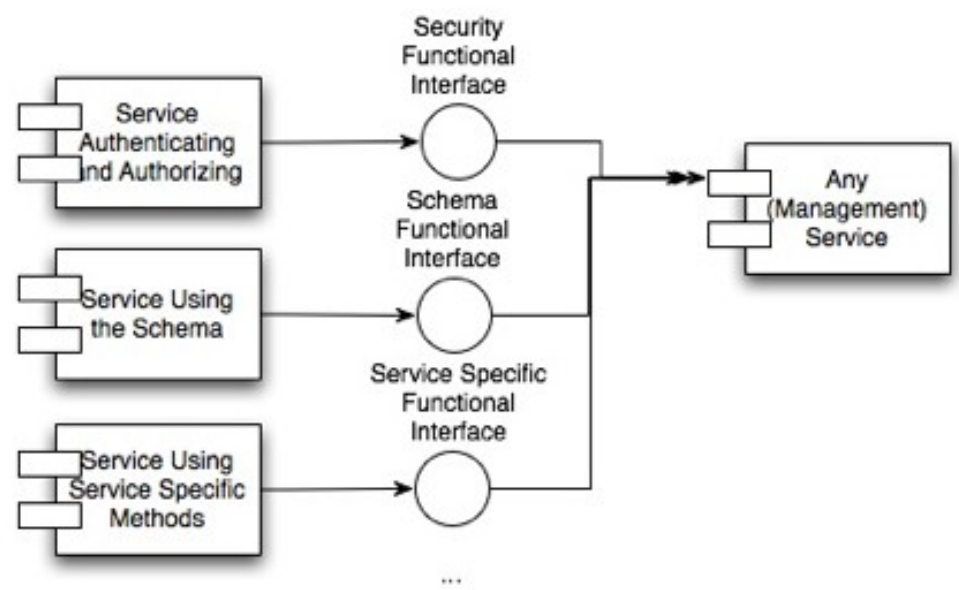
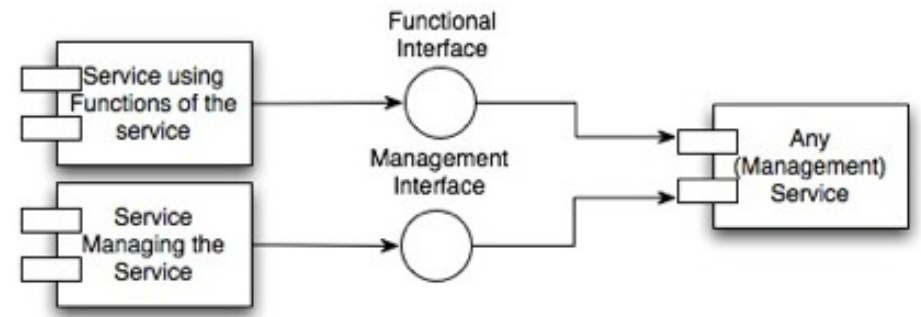


Decomposition of Service Interfaces

A service's interface breaks down into a functional interface (access to the service's functions) and a management interface (access for the purpose of managing its operations)

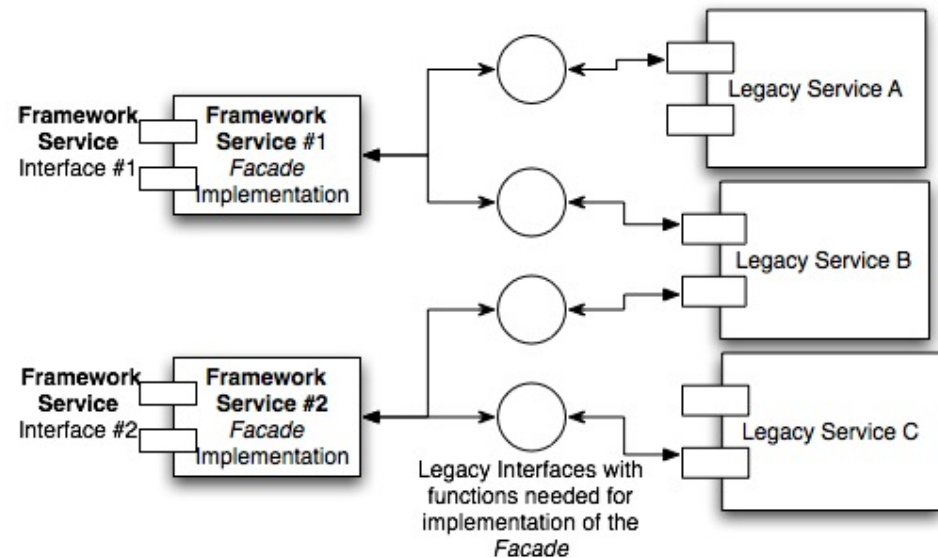
A service's functional interface can be broken down into any number functional interfaces that represent different aspects of its functions

Polymorphism of interfaces allows a service to implement multiple of these



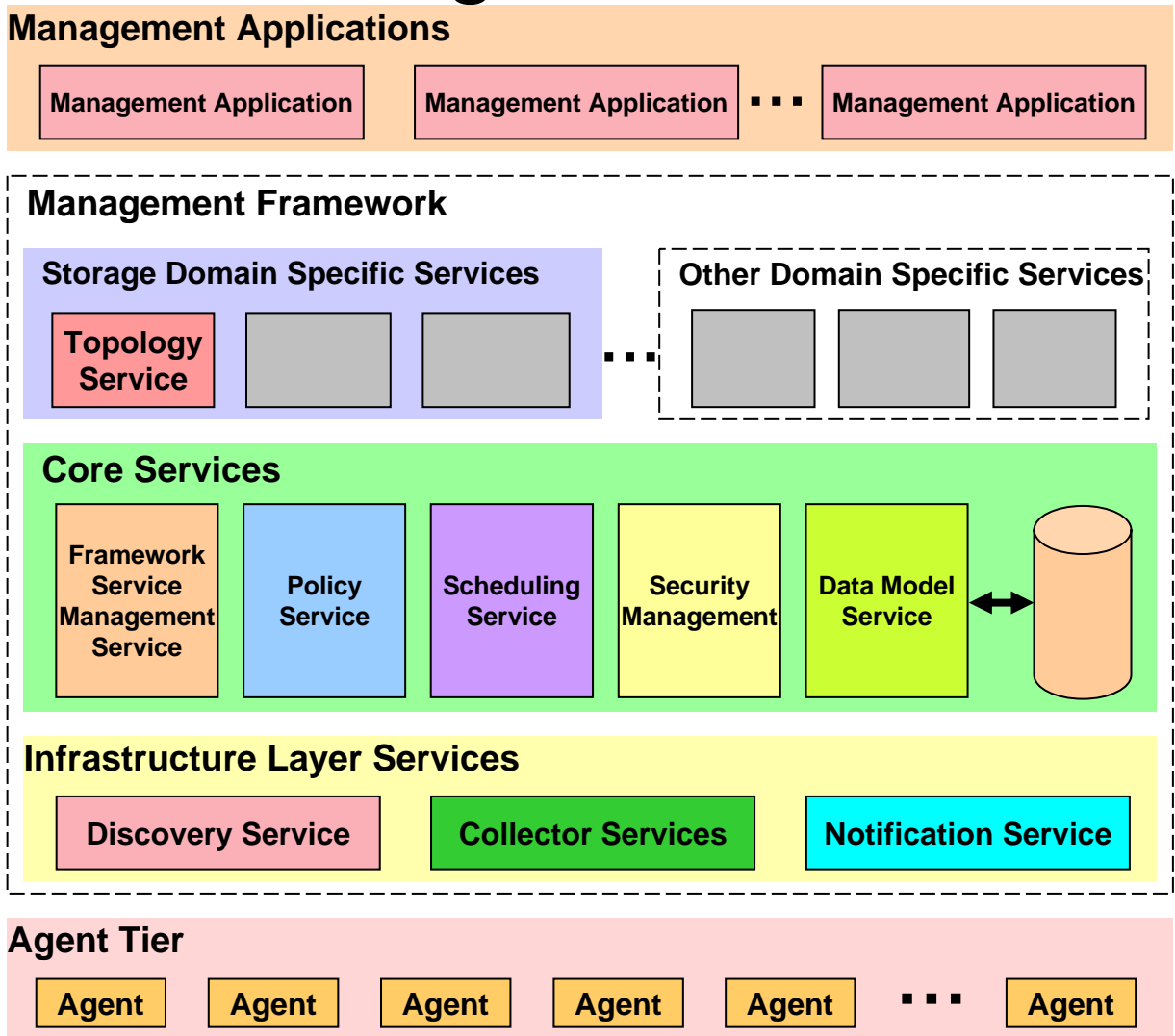
Dealing with Legacy Frameworks

- How do existing frameworks support the new standard?
- A Framework Service Façade can wrap functions of the legacy services and present a standard interface to applications and other services



Later, the framework can be re-factored to present the standard interfaces directly

High Level Architecture



The layers of the architecture include:

- Infrastructure Layer Services
- Core Services
- Domain Specific Service

Layers

- Infrastructure Services
 - Interact with managed element agents for discovery, for collecting management data, and for subscribing to asynchronous notifications
 - Can be used by any other service in the management framework
 - Can grow to support new protocols and capabilities in the future
- Core Services
 - Are not specific to any particular management domain, and are separated from the Agent tier by the Infrastructure Services
 - Includes services that are fundamental to any management application and provide an adequate base for building such applications
- Domain Specific Services
 - Extend the Core Services layer with additional services that are fundamental to handling a specific management domain

Common Aspects of Services

- Common aspects of services are those functions that all management framework services need to implement
 - Functional Interface
 - Management Interface
- Common aspects of services include:
 - Security
 - Service Discovery
 - Runtime Management
 - Lifecycle Management

Common Security Aspect

- All management framework services are expected to handle security concerns in a common fashion
 - Authentication, Authorization, and Audit functionality
- The principle of least privilege applies for all interactions with and between management framework components
- Must not introduce any security vulnerabilities with the management framework itself
- Security Management Service (in Core Services layer) provides a means of managing the common security aspects of framework services
 - There is exactly one Security Management Service in a Management Framework Instance (MFI)

Other Common Aspects

- Discovery
 - All management framework services must support service discovery in a common fashion
- Runtime Management
 - All management framework services must support a common set of runtime management functions
 - Runtime environment may also be managed to prioritize services, understand load, balance load, etc
- Lifecycle Management
 - All management framework services must support a common set of lifecycle management functions
 - For installing, upgrading, migrating, decommissioning, etc

Infrastructure Services

- 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.

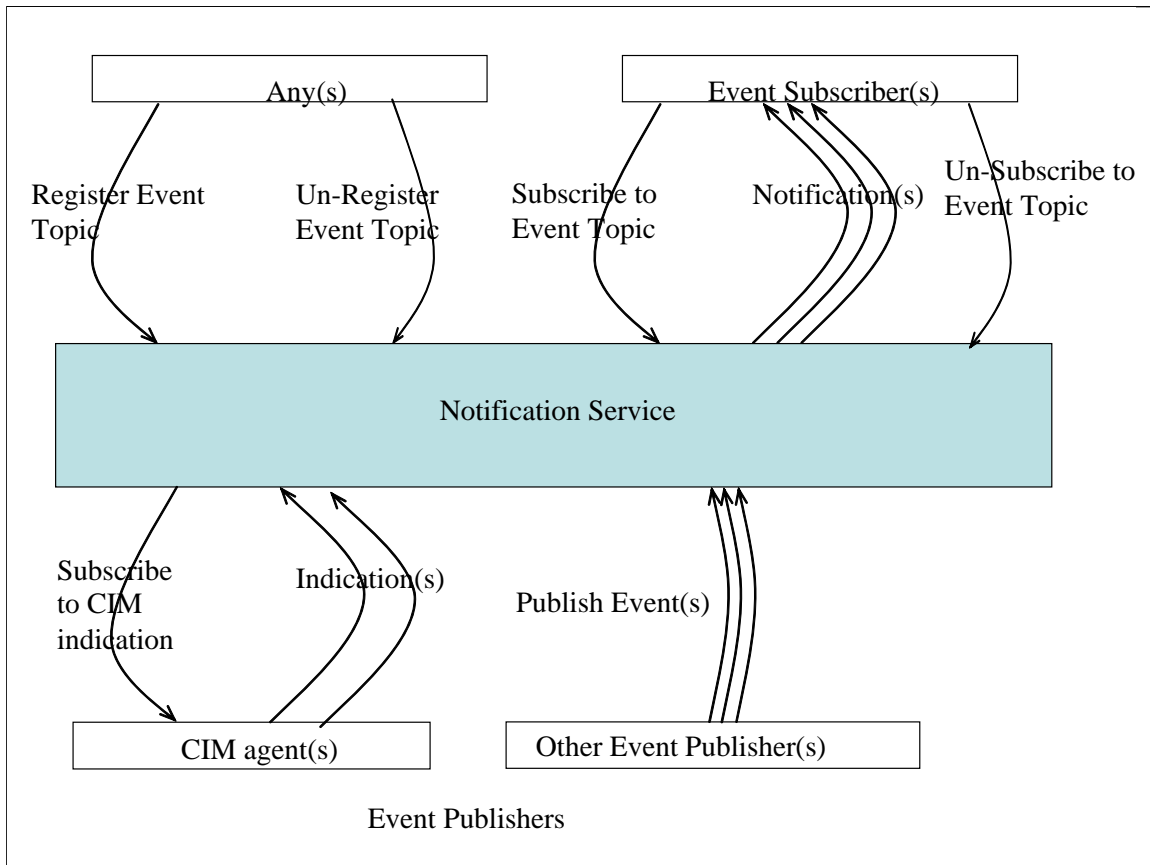
Collector Service

- 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*

Notification Service Characteristics

Any component of the management environment may use the Notification Service

Characteristics:

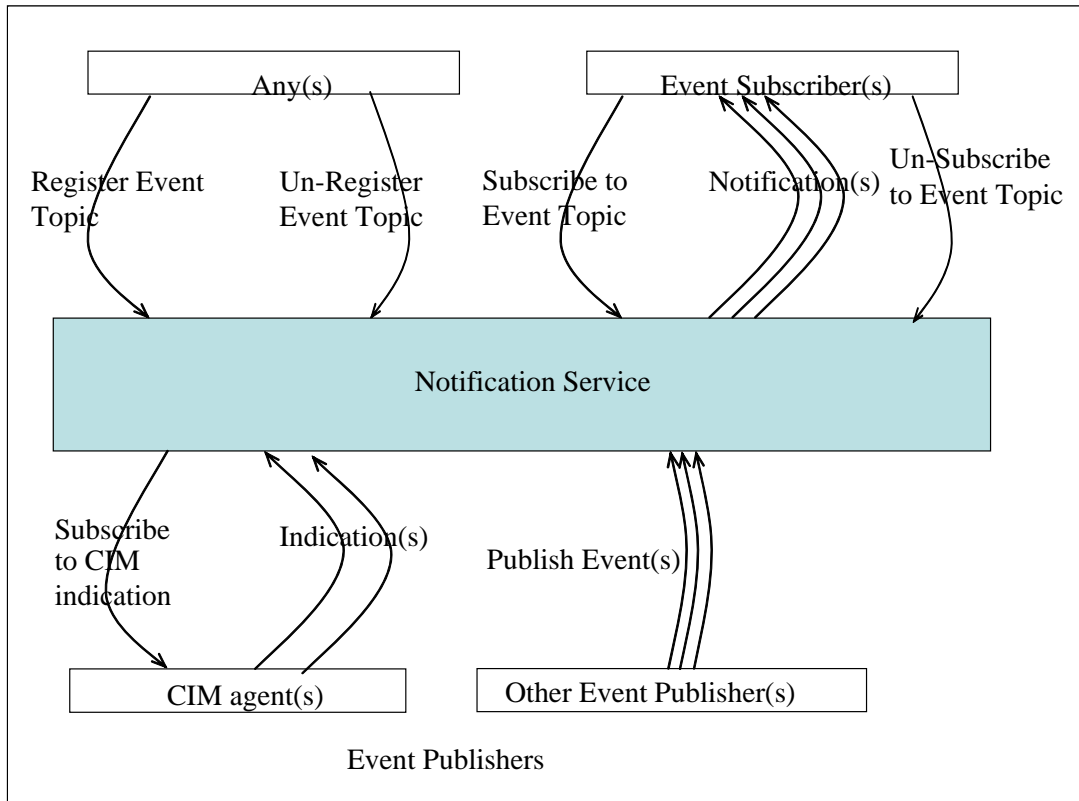
Anonymous

Asynchronous

One-way

One-to-many

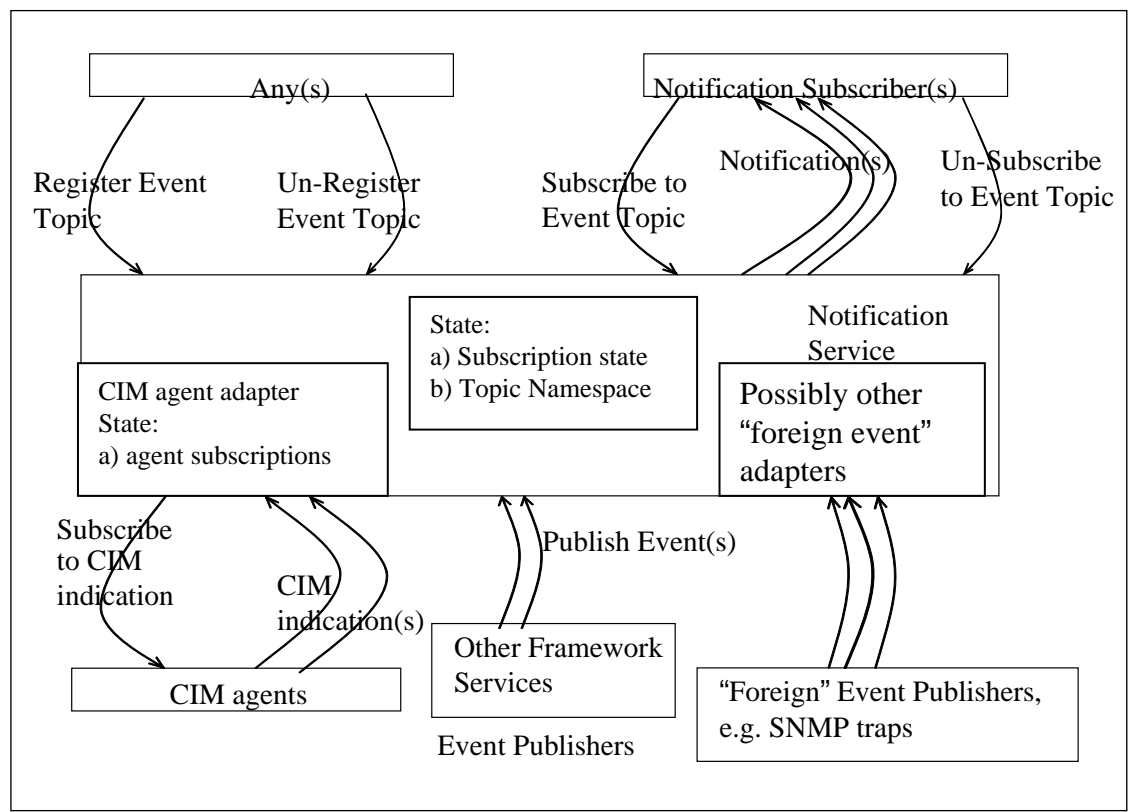
Unsolicited



Exactly one Notification Service per MF Instance

Notification Service Detail

Integration with event sources, including CIM indications

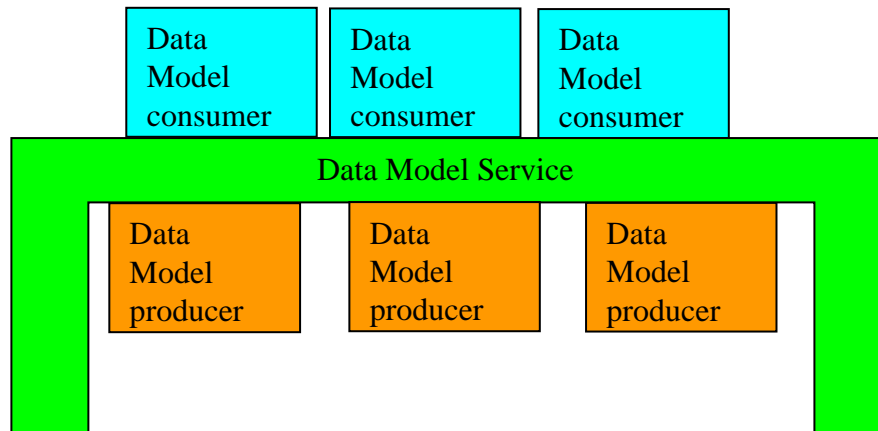


- Subscriber need not be aware of the source of an event
- Topic Namespace includes CIM indications
- Subscribing to an Event Topic triggers a CIM indication subscription
- Other event sources can be similarly integrated
- Events can be published directly

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

Data Model Service



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

Framework Service Management Service

- Framework Service Management Service provides a means of monitoring and controlling key aspects of how the framework and its services use the execution environment.
- Provides a standardized abstraction for this purpose
 - To accommodate various execution environments.
- First service to start up when a Management Framework Instance (MFI) is “booting up”
 - It starts up all of the other MFI components
 - It uses the Data Model Service to maintain a model of the MFI
 - But must be able to handle the first-boot situation after installation
 - Two cases for booting up an MFI
 - First boot up after installation: must bring up initial configuration
 - All other boot up cycles: must bring up (or continue) the MFI configuration that was last true
 - It determines where the MFI components are instantiated
 - May rely upon the underlying execution environment to determine where

Framework Service Management Service - continued

- There is exactly one FSM per management framework instance
 - It must be present and therefore should be reliable/redundant
 - Must handle the case where the FSM comes back and the rest of the MFI is already running
- It monitors all of the other MFI components for failures
 - Tries to re-start any MFI components that fail

Scheduling Service

- The Scheduling Service provides the ability to initiate actions in other management framework services...
 - via their defined interfaces,
 - on a persisted schedule.
 - Ignores returned status from action initiation attempts.
- Provides economy and consistency of implementation...
 - Avoids redundant function in individual services.
- Typical uses of this service include invocation of Discovery and Collector services.
- The Scheduling service persists schedule information in the Data Model Service as CIM object instances.

Policy

- 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.

Topology Service

- 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.

How will this be deployed?

- Even if the framework services are deployed across multiple (real or virtual) machines, it is still (logically) one MFI
- Scaling is done through multiple MFIs
 - Each responsible for some “scope”
 - Partitioned among domains, resources
- Domain specific (non-framework) services can either be aware of multiple MFIs or not
 - Depending on their scope

Questions?