HARVARD
UNIVERSITY

Information Technology

Integer - An Integrated Management Environment
Design Review - v1.0

June 25, 2014

Topics

- Introduction - Jim Waldo.
- Problem.
- Architecture.
  - Implementation technologies.
  - Logical design.
  - APIs/Interfaces.
  - First release services.
  - Object hierarchies.
- Deployment.
- First release - implementation and operation.
- Contact us and information.

Problem

What’s the Problem?

- We don’t have a service based view of our systems and services.
- Information gaps cause downtime:
  - Isolated management systems/approaches ‘databases’.
  - Absent/uncoordinated monitoring across environments.
- Patchwork of non-integrated management systems from multiple sources is not cost-effective:
  - Multiple groups doing small, sometimes overlapping development.
  - Problems with access to timely information.
  - Duplication of effort.
  - Can increase time to identify and resolve problems.
  - Makes service provisioning and de-provisioning more complex, error-prone and less agile.
- New environments like AWS add complexity and proprietary methods further fragmenting our view; more stove pipes.

Service Outage Analysis

- No service configuration software - software that understands the relationships between all the elements of a service.
- Installation and upgrade procedures require significant human interaction, increase error opportunities.
- Effective root cause/systems failure analyses are not routinely performed - do not have data and systems that facilitate such analyses.
- Unknown service dependencies - 74% of a key service's outage time was the result of unknown dependencies. We did not understand how a change (or a fault), on apparently unrelated systems would impact the service.
- Configuration error - configuration error is a leading cause of failure in the industry. Harvard still has a largely manual configuration process - many steps can introduce errors.

Integer's Scope

- Covers essential areas of management with a whole-service view:
  - Configuration:
    - Automated and reliable configuration of systems and services - including deployment of systems & images.
  - Performance:
    - Issues related to latency and capacity.
    - User level to fine-grained per-system details.
  - Fault - detecting and repairing failures.
  - Security - Service wide perspective security controls.
  - Accounting - quantity of work done on behalf of a service.
- Key elements of the environment:
  - Users.
  - Technical Components:
    - Servers.
    - Network elements like routers, firewalls, load balancers, DNS system and other physical and virtual network elements.
    - Software from the virtualization layer to high-level web services.

Integer’s Primary Differentiator

- Works with network devices, servers and software as they exist (i.e., no modifications required).
- Accepts reality of multiple monitoring and control protocols per system.
- Does not assume managed systems adopt CIM, WebM or any other model or current fad.
- Creates a common representation from these disparate protocols:
  - Enables a services view.
  - Allows cross vendor and technology configuration.

Why Start with Discovery

- Necessary to have up to date information of systems/services being managed.
- Consensus that it was time to upgrade our existing discovery system.
- No discovery currently provides linkage between network layers.
- Allowed us to build enough of framework to validate architecture and deliver user value in relatively short time.
- Allows early feedback.

Architecture

Architectural and Implementation Focus

Integer is designed to view, manage, and understand the relationships of all the elements of a business service as a whole.

System Overview

- Web front end.
- A set of services with APIs that hide DB structures.
- Coded in Java.
- A ‘core’ installation that can be clustered/distributed:
  - Reliability.
  - Performance.
- Multiple distributed server installations.
  - To reduce network impact.
  - Reduce network configuration complexity.

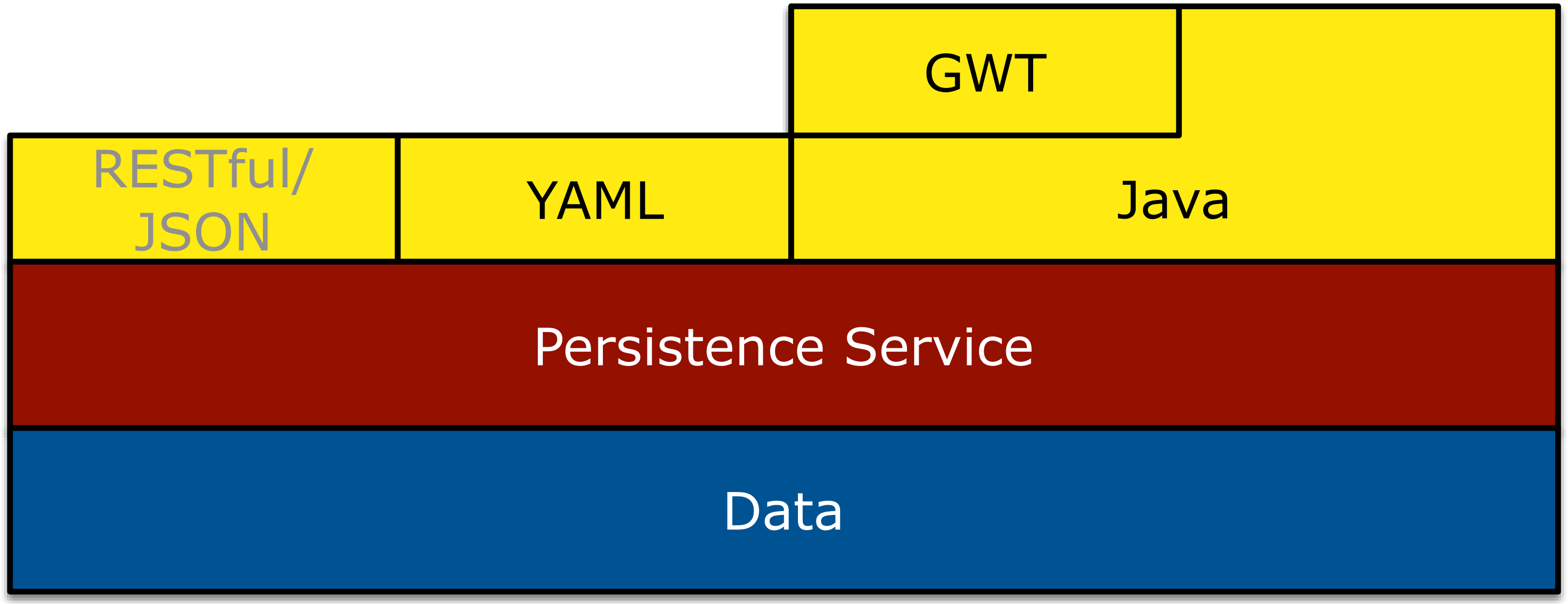
Open Source Technologies/Components

Database	MariaDB
NoSQL	Cassandra - not used in initial discovery release
Middleware	WildFly (a.k.a. JBOSS)
UI	GWT - Google Web Toolkit
Scheduling/Time	Quartz
SNMP	SNMP4j
MIB Compiler/ Loader	Mibble
Logging	SLF4J - Simple Logging Facade for Java

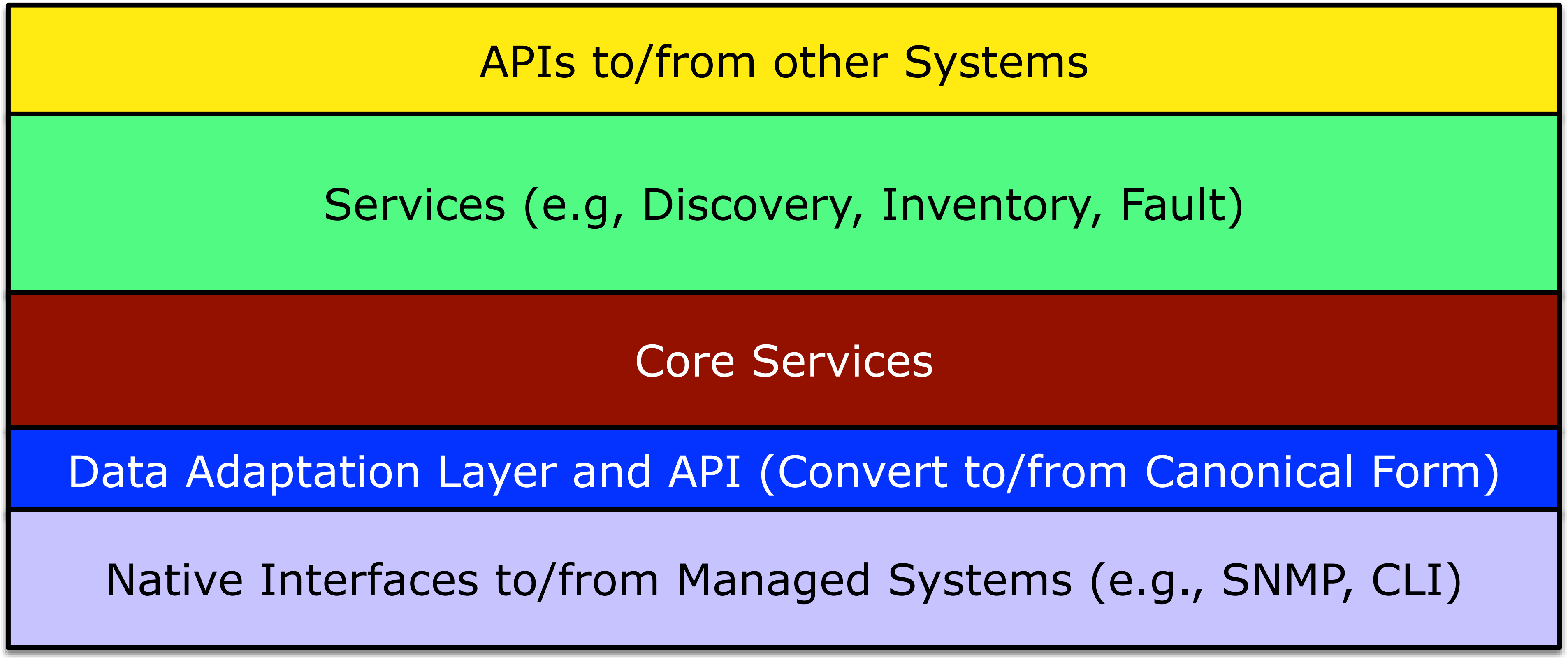
Components We Are Developing

- Service that accesses data in the DB(s)
  - DAO’s.
  - APIs to our objects.
- Using WildFly - code to control distribution of our system.
- Interfaces to native interfaces used by devices, e.g., discovery service uses open source SNMP code.
- Our business logic/services, major elements:
  - Discovery
  - Inventory
  - Reporting
- UI built out of GWT with some custom JS.

Integer Data and API- Centric View



Server Logical Design



APIs to/from other Systems

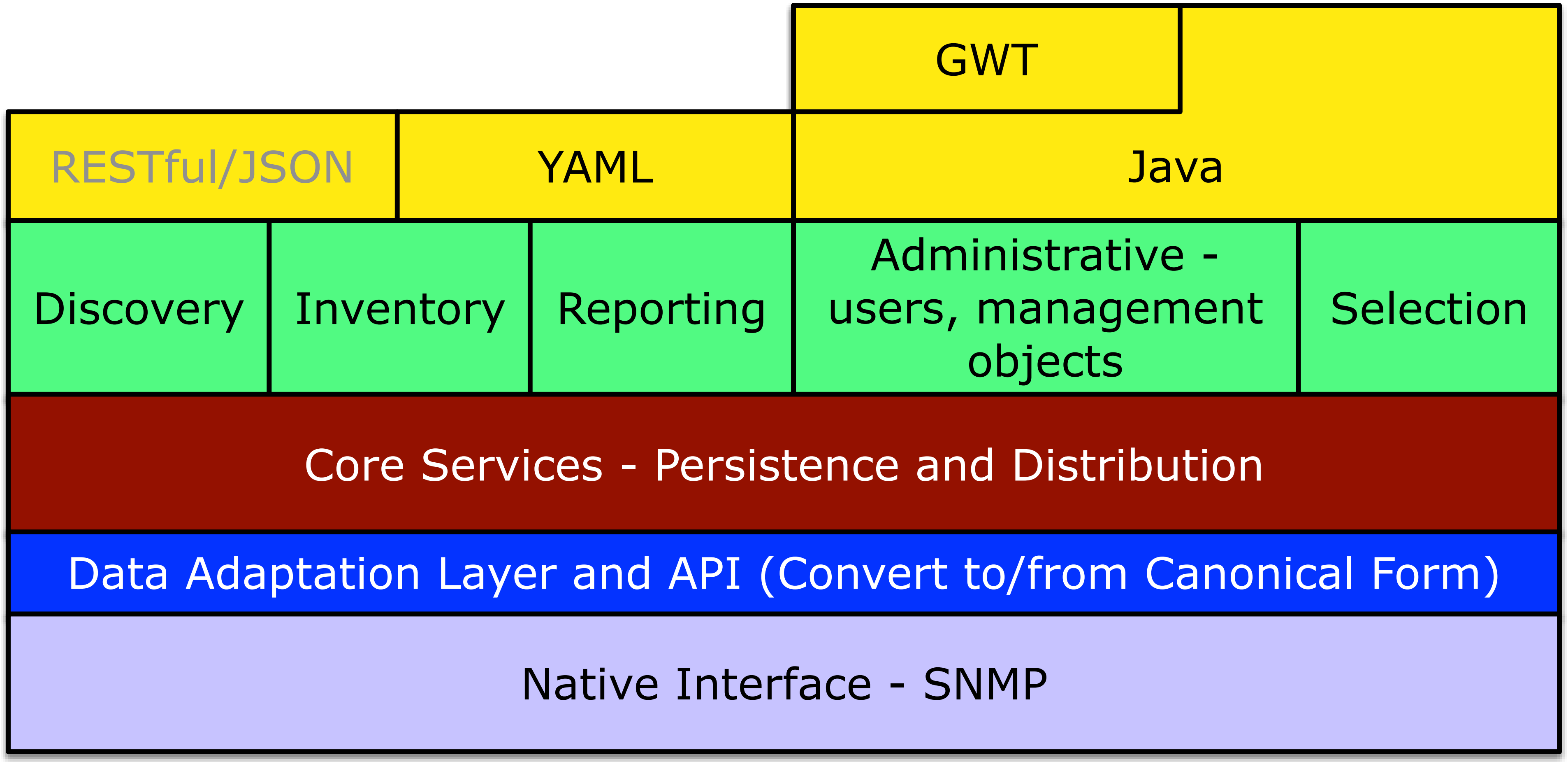
Services (e.g, Discovery, Inventory, Fault)

Core Services

Data Adaptation Layer and API (Convert to/from Canonical Form)

Native Interfaces to/from Managed Systems (e.g., SNMP, CLI)

Discovery Release - APIs, Services, Data Adaptation and Native Interfaces



GWT

RESTful/JSON

YAML

Java

Discovery

Inventory

Reporting

Administrative - users, management objects

Selection

Core Services - Persistence and Distribution

Data Adaptation Layer and API (Convert to/from Canonical Form)

Native Interface - SNMP

APIs and Interfaces

- Programer Centric
  - GWT
  - Java
- User-based
  - RESTful/JSON
  - YAML
- Others based on user input.

Integer Services and Managers

- Services are singleton beans.
- Managers - stateless session beans.
- As many managers of a specific type as are needed/ configured, e.g.,
  - Topology.
  - Service Element Discovery.
- Aids scaling.

Core Services - Persistence

- All access to Integer data.
- Access only via service, no direct DB access - hides future changes from users and developers.
- Access implemented with DAO’s on top of Hibernate - common technique to isolate storage/db technology from application writers. Allows code reuse for querying database.
- Storage via MariaDB - Open source, solid, good performance, reliable, free.
- MariaDB has Cassandra storage engine - allows single interface to both systems.
- Cassandra will store stats. Data serialization in and out of DBs is often problem with NM systems.
- Galera cluster for synchronous multi-master MariaDB.
    - Active-active multi-master.
    - Read and write on any node.
    - Can operate across data centers.

Core Services - Distribution

- Distribution service for systems and functions they perform.
- Keeps track of services and distributed elements.
- Each system is separately configured.
- Keeps track of state and receives messages from distributed systems.
- Uses native WildFly mechanisms for messages.

Discovery Service/Managers

- Service controls:
  - Start/stop of service element discovery.
  - Start/stop of topology discovery.
  - Per sub-net/discovery rule.
- Processing ‘rules’
  - Discovery rule
  - ipTopologySeed
  - Global credentials
  - snmpLocalCredentials
  - CalendarPolicy
- State/status of running discovery.

Inventory Service

- Controlled by inventory rules:
  - What service elements are covered by the rule:
    - Location.
    - Criticality.
    - Technologies (e.g, routers, name servers, etc.).
    - Changes since last run.
    - New systems.
    - Missing systems.
  - Definition of:
    - What it means to be ‘missing’ - number of runs absent.
    - Number of runs a new system must exist before being added.
    - Notification actions.
    - Systems to exempt (e.g, those with ifAdmin status down).
    - What a change is for a specific type of service element (based on unique identifiers for service element types).

Reporting Service

- Same selection method as other parts of the system.
- For this release:
  - Entire inventory.
  - Changes or other selections.
- Basic format/text report.
- JSON output.
- Email.
- Event/alarm window.

Selection Service

- Same approach for interacting with Integer for all functions, e.g,:
  - What you want to see in a report.
  - What you want to see on the screen.
  - What to discover.
- Elements of a selection
  - The systems/services.
  - Should topology information be included.
  - The information about the selected systems/services of interest
    - State and (configuration, discovery, fault, etc.).
    - Utilization and other historic information.
    - Capacity.
    - Calculated values input by users.
  - How to present the information, e.g,
    - As a topology map.
    - A tabular report.
    - Output to CSV or JSON.

Other Services/Managers

- Events/Alarms and logs.
- User and roles.
- Information for the data adaptation layer.

Data Adaptation Layer

- Maps between device and protocol details to Integer.
- Data driven.
- User modifiable.
- Consistent user experience.

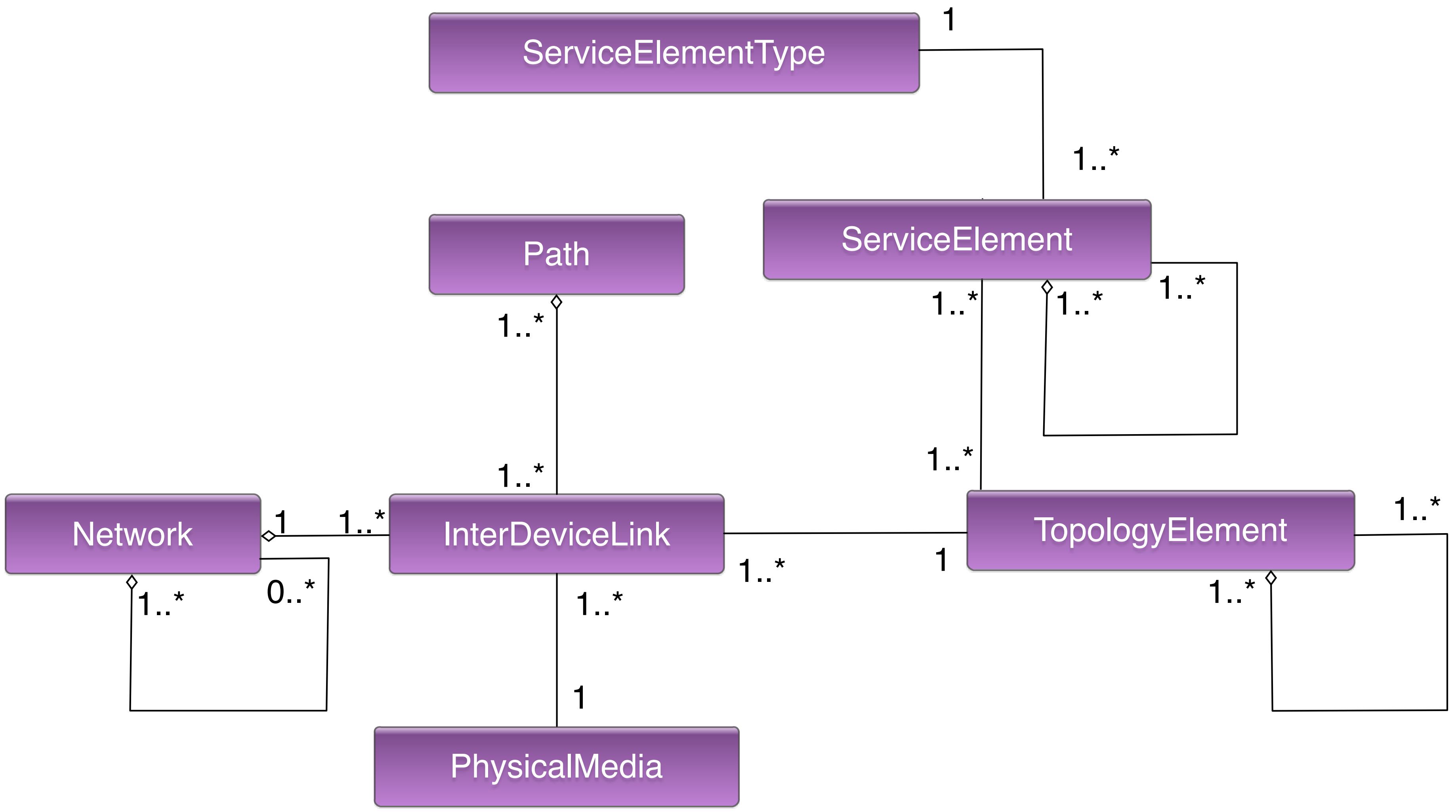
Object Hierarchies

- Main conceptual elements
  - Business services and technologies.
  - Providers and organizations.
  - Users, groups, roles.
  - Service elements and multi-layer/technology topology.
- Others used for various functions and services:
  - Discovery.
  - User selections.
  - Reporting.

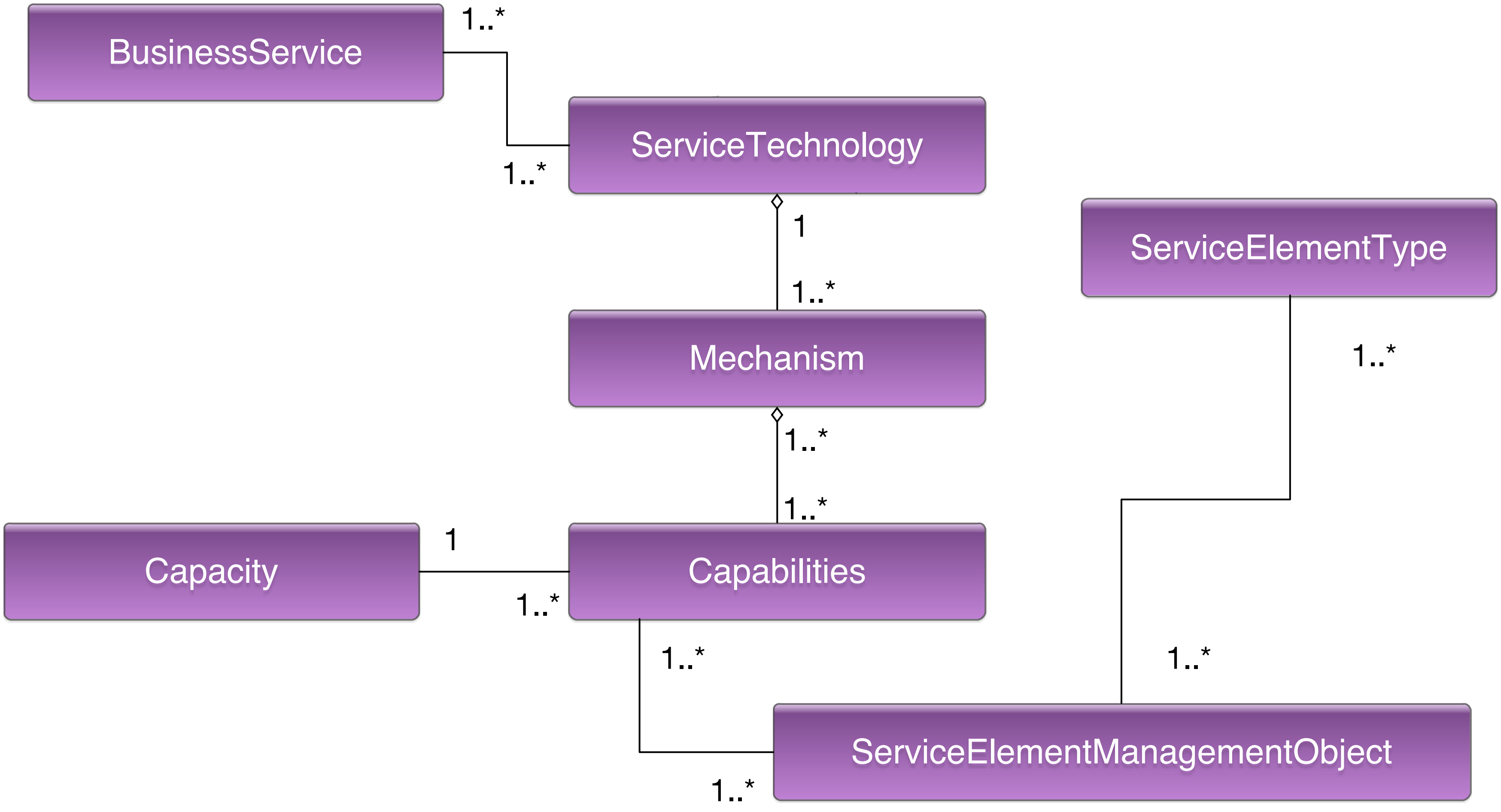
Integer Domain Knowledge

- How Integer ‘learns’ about and normalizes to a canonical form, different types of management objects:
  - SNMP - standard and vendor-specific.
  - A variety of CLIs.
  - RESTful and other interfaces and data formats used on them
  - Different APIs/interfaces for different environments like AWS.
  - DevOps and automation systems like Puppet, Chef, etc.
- How Integer ‘learns’ what makes a particular device, or part it contains unique in terms of what it can do.
- How Integer connects the large number of combinations into knobs that a human can control and understand and use to train Integer further.

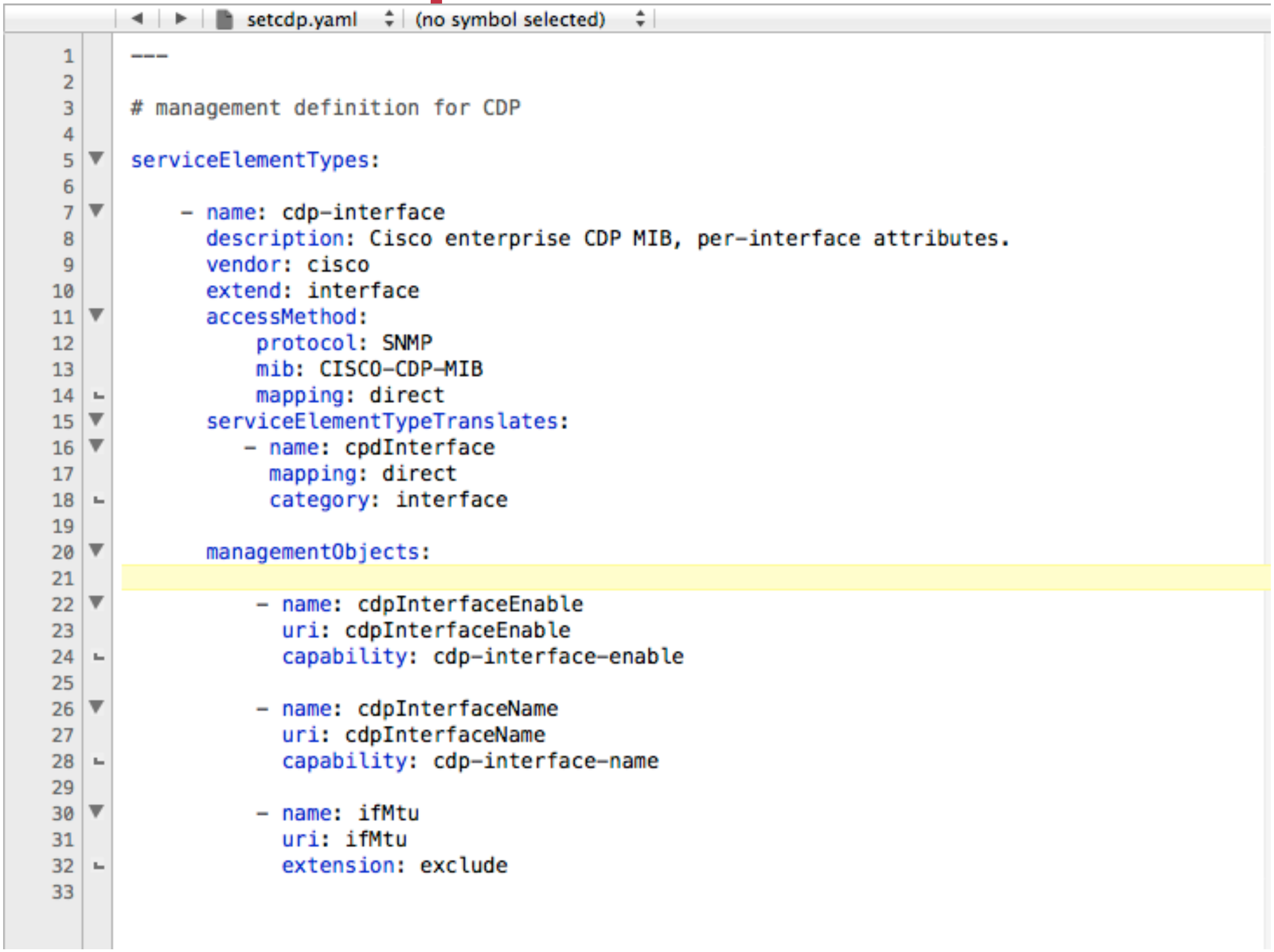
Systems and Topology



Domain Knowledge Enables Service Management



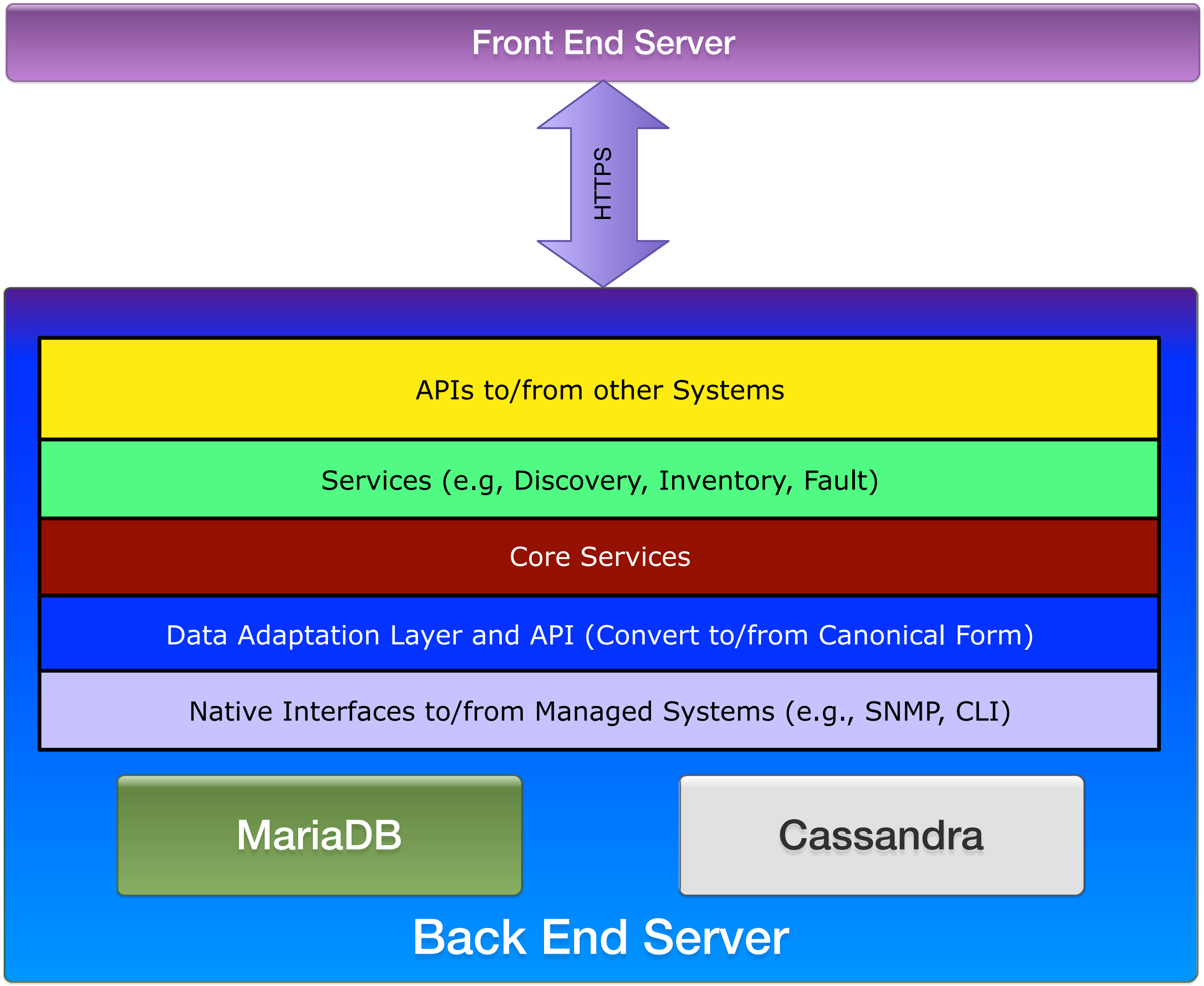
YAML Example



```yaml
---

# management definition for CDP

serviceElementTypes:

    - name: cdp-interface
      description: Cisco enterprise CDP MIB, per-interface attributes.
      vendor: cisco
      extend: interface
      accessMethod:
          protocol: SNMP
          mib: CISCO-CDP-MIB
          mapping: direct
      serviceElementTypeTranslates:
         - name: cpdInterface
           mapping: direct
           category: interface

      managementObjects:

          - name: cdpInterfaceEnable
            uri: cdpInterfaceEnable
            capability: cdp-interface-enable

          - name: cdpInterfaceName
            uri: cdpInterfaceName
            capability: cdp-interface-name

          - name: ifMtu
            uri: ifMtu
            extension: exclude

```

Deployment and Operation

Distribution - Simple Deployment



Front End Server

HTTPS

APIs to/from other Systems

Services (e.g, Discovery, Inventory, Fault)

Core Services

Data Adaptation Layer and API (Convert to/from Canonical Form)

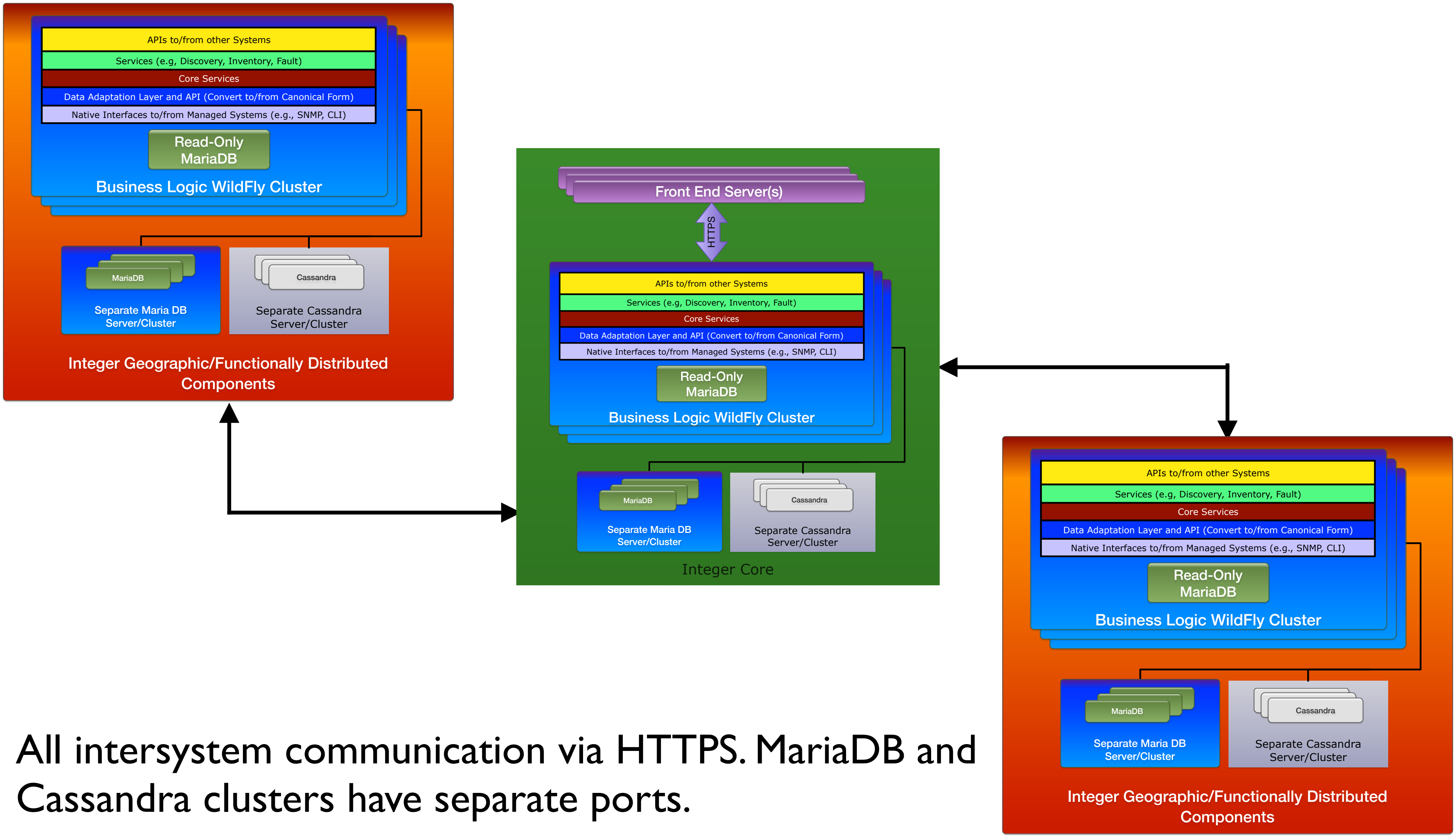
Native Interfaces to/from Managed Systems (e.g., SNMP, CLI)

MariaDB

Cassandra

Back End Server

Distribution - Large/Complex Case


APIs to/from other Systems
Services (e.g, Discovery, Inventory, Fault)
Core Services
Data Adaptation Layer and API (Convert to/from Canonical Form)
Native Interfaces to/from Managed Systems (e.g., SNMP, CLI)
Read-Only MariaDB
Business Logic WildFly Cluster
MariaDB
Separate Maria DB Server/Cluster
Cassandra
Separate Cassandra Server/Cluster
Integer Geographic/Functionally Distributed Components
Front End Server(s)
HTTPS
APIs to/from other Systems
Services (e.g, Discovery, Inventory, Fault)
Core Services
Data Adaptation Layer and API (Convert to/from Canonical Form)
Native Interfaces to/from Managed Systems (e.g., SNMP, CLI)
Read-Only MariaDB
Business Logic WildFly Cluster
MariaDB
Separate Maria DB Server/Cluster
Cassandra
Separate Cassandra Server/Cluster
Integer Core
APIs to/from other Systems
Services (e.g, Discovery, Inventory, Fault)
Core Services
Data Adaptation Layer and API (Convert to/from Canonical Form)
Native Interfaces to/from Managed Systems (e.g., SNMP, CLI)
Read-Only MariaDB
Business Logic WildFly Cluster
MariaDB
Separate Maria DB Server/Cluster
Cassandra
Separate Cassandra Server/Cluster
Integer Geographic/Functionally Distributed Components

All intersystem communication via HTTPS. MariaDB and Cassandra clusters have separate ports.

Discovery Release Objectives

- Covers SNMProwl +.
- Complete inventory of the environment.
  - Network elements - including layer 2 and message stream modification systems like:
    - Firewalls.
    - NAT.
    - Load balancers.
  - Servers.
  - Interconnections.
  - Details of the ‘contents’ of each system - e.g., cards, ports, etc.
  - Some high-level software.
- Automated system for detecting, tracking and reporting changes to systems from the hardware up (expands over time).
- Feed to systems like Nagios that require ‘inventory’ information.

Discovery Deployment

- Can be deployed as a single system or in a distributed fashion
  - Each system can be configured in terms of the scope of subnets included in the discovery.
  - Each of the distributed elements rolls up to the main installation.
  - Each can have a separate set of ‘rules’ that control behavior.
- Systems can run in virtualized, physical or cloud-based environments or any combination.
- Each ‘rule’ may have a different schedule if desired.
- Each system may have multiple tasks discovering devices and topology.

SNMP Background

- SNMP - Simple Network Management Protocol
- Standard - don’t have to implement any low level protocols
- We don’t have to add software to systems, SNMP is widely implemented on servers, middleware, databases, routers, switches, load balancers, etc.
- Familiar to operations people.
- Strengths and weaknesses understood.

SNMP High-Level Details

- Multiple versions of the framework in use, primarily
  - SNMPv2
  - SNMPv3
- Data
  - Stored in a Management Information Base - MIB.
  - Stored as single instance objects or in tables.
- Language for defining objects:
  - Structure of Management Information (SMI).
  - “Imperfect” subset of Abstract Syntax Notation.1.
- Protocol:
  - 3 Query functions.
  - One ‘set’ function.
  - Two functions for asynchronous messages ‘notifications’.
- Administrative framework.
    - Clear text community strings.
    - SNMPv3 added provisions for authentication and privacy (MD5, SHA, DES).

Discovery Mechanism

- First release is exclusively SNMP-based.
- Find devices that will respond to SNMP requests.
  - SNMP version.
  - Community or v3 Administrative information.
  - Port.
- Identify vendor and other information (vendor/system specific).
- Use information to identify device containment hierarchy (e.g, proprietary, entity MIB, Host Resources, SysAppl, etc.).
- Identify each of the children.
- Retrieve interface information via ifMib and extensions.
- Assigns a ServiceElementType based on unique signature.
- Create systems and associated network information for found interfaces.

Discovery - Service Element Discovery

- Determine range of hosts to check on each subnet that has been assigned - based on subnet/mask or CIDR address.
- For each system found in range we attempt to retrieve 6 elements from the System Group we use to:
  - Establish vendor and top level information such as software version.
  - Determine device's containment structure (e.g., cards, ports, memory, storage, etc.
  - Learn about each sub-element (types, whether it contains sub-elements, etc.).
  - Learn about interfaces and network connections and addresses.
- Based on discovery configuration, learn about other technologies from routing to load balancing currently enabled on the service element.
- Save to database.
- Repeat for each network specified in configuration (or learned from topology discovery - next slide).

Discovery - Topology

- Use seed information from Discovery Rules.
- Information from device discovery (if run previously).
- Discovery of different technologies relevant to topology and connectivity:
  - CDP.
  - LLDP.
  - Routing information by protocol if desired:
    - Static.
    - BGP.
    - OSPF.
    - RIP.
    - Etc.
  - Other types of interconnection such as router or other types of redundancy.
  - Layer 2 (VLANS).
  - Layer 2.5 like MPLS.
- Discover additional networks based on rules.
- Save information to database.

Discovery - Next Steps

- User interface for domain knowledge.
- Manual insertion/placement of layer 1 elements.
- Extend to service dependencies.
- Extend to storage.
- Extend to cloud (e.g., AWS)
- Extend to virtualization.

integer.harvard.edu

How You can Participate

- info@integer.harvard.edu - specific requests for information or to make comments.
- integerinfo@integer.harvard.edu - mailing list used periodically to send out news and updates on the initiative.
- integerproject@integer.harvard.edu - a subscription mailing list for people to discuss the initiative.
- Become a ‘tester’.
- Contribute other skills and knowledge.
- publictest.integer.harvard.edu.