OOI CYBERINFRASTRUCTURE
NSF OCEAN OBSERVATORIES INITIATIVE

OOI – CyberInfrastructure

Architecture & Design

Systems and Services Interface Description

SV-1 PDR CANDIDATE

November 2007

NSF JOI Joint Oceanographic Institutions UCSD

Document owner: OOI CI Design Team

Document History

Date	Version	By	Description of Changes
2007-08-03	1.0	CI ADT	Preliminary Network Design Initial Draft
2007-10-03	1.1	CI ADT	Updated
2007-11-12	1.2	CI ADT	Merged with SV-2 contents, reorganized

Disclaimer: All architecture documents are intended to be living documents so as to reflect changes to requirements and their design impact.

Preamble

The set of documents named AV*, OV*, SV*, TV* are all part of the OOI CyberInfrastructure Architecture & Design (CIAD), in the structure prescribed by the DoDAF (Department of Defense Architecture Framework). Each document has a designated title, an identifier (such as AV-1) and covers a specific topic in a self-contained way. Document AV-1 provides further explanations and a summary. A glossary of the terms used in these documents and their context can be found in AV-2.

The figure below suggests an intuitive reading flow through the provided documents. Other documents will be added to the figure as they emerge during the design of the CI (for the complete set of documents see AV-1). The thick arrow suggests a reading order through the core documents (AV-1, OV-1, OV-5 and SV-1). The red rectangle highlights the current document.

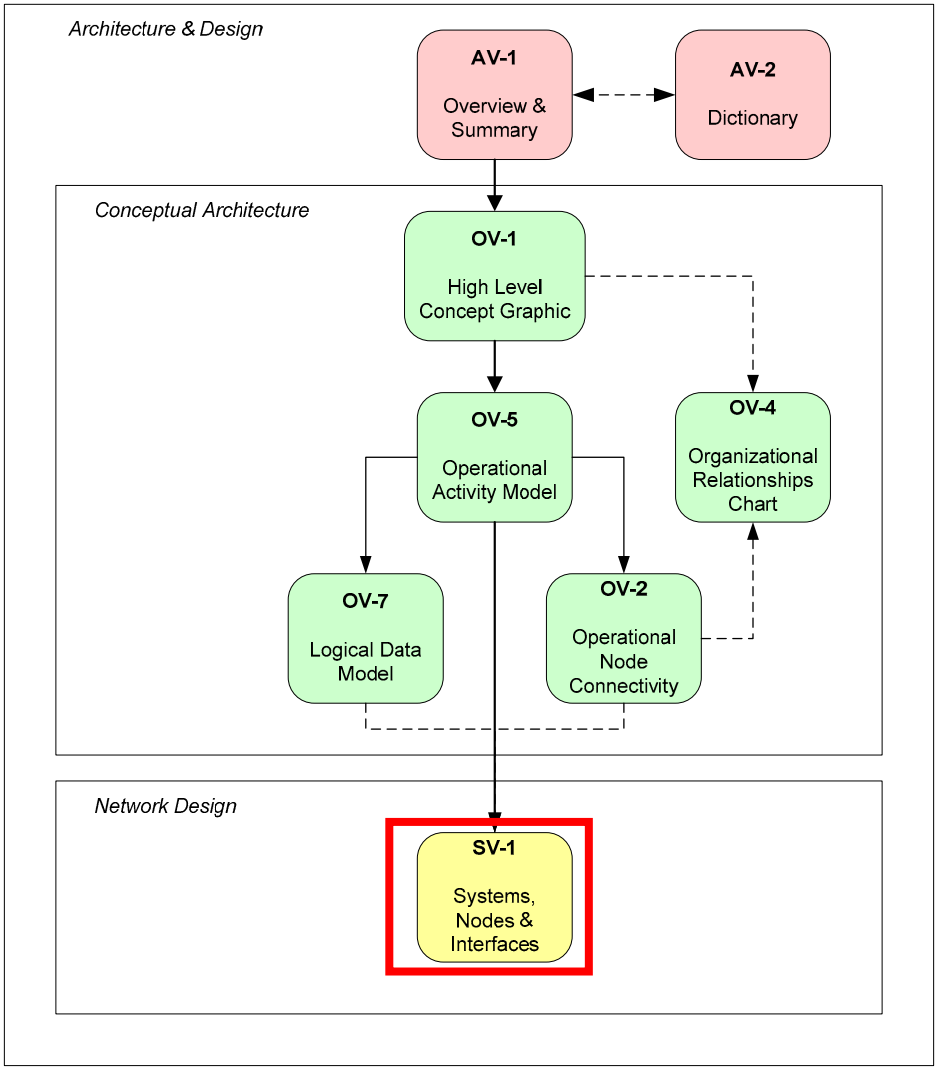

Table of Contents

OOI - CyberInfrastructure
Architecture & Design
Systems and Services Interface Description (SV-1)

1 Introduction

1.1 Product Overview

This document depicts systems nodes and the systems resident at these nodes to support organizations/human roles represented by operational nodes of OOI as represented in the OV-2. It also identifies the interfaces between systems and systems nodes (adapted from [DoDAF-vII 2007]).

1.2 Product Purpose and Description

The goal of this document is to identify systems nodes and systems that support operational nodes and to link together the OVs and SVs by depicting the assignments of systems and systems nodes (and their associated interfaces) to the operational nodes (and their associated needlines) described in OV-2.

OV-2 depicts the operational nodes representing organizations, organization types, and/or human roles, while SV-1 depicts the systems nodes that house operational nodes (e.g., platforms, units, facilities, and locations) and the corresponding systems resident at these systems nodes that support the operational nodes. Only systems, subsystems, or hardware/software items and their associated standards are documented here, where applicable.

Details of the communications infrastructure (e.g., physical links, communications networks, routers, switches, communications systems, satellites) are documented in the Systems Communication Description (SV-2). An SV-1 interface is the systems representation of an OV-2 needline (adapted from [DoDAF-vII 2007]).

2 CyberInfrastructure Technologies

2.1 Basic Communication Technologies

2.1.1 Message-orientation

The communication system of the OOI CyberInfrastructure bases on messaging as the central paradigm of inter-application information exchange. Message-oriented systems and message-oriented middleware (MOM) technology realize a system architecture pattern that enables the flexible integration of systems of systems while retaining a loose coupling. Loose coupling is one of the most important architecture criteria that positively influence maintainability, extensibility, robustness, scalability and other quality properties of the system and its individual software components.

Messaging infrastructures provide the concept of a message as exclusive means of information exchange between the distributed components of the system. All information that is passed between two components is contained in the exchanged messages. Message exchange is asynchronous, i.e. the sender of a message does not wait for the message to be delivered or returned. It only waits for the MOM to acknowledge the hand-over of the message. Addressing messages to recipients is based on the concept of queues. An application component in a message-oriented architecture only knows the incoming queues that it receives messages from as well as the outgoing queues it delivers messages to, plus the message formats that pertain to these queues. The messaging infrastructure provides the capabilities for system integrators to connect these queues to known endpoints in the network; consequently it handles routing,

storing and delivering the messages to the intended recipients across the network. The concepts of message exchange as the only means of inter-component communication, of logical, configurable queues and of message storing and routing are the main enablers for loose coupling of distributed components, for instance in wide-area network connected, federated systems with intermittent network connections, such as the OOI CI.

Messaging infrastructures typically provide two styles of addressing messages: point-to-point and publish-subscribe style of messaging. The first case connects one sender with one receiver through a queue and thus decouples both components while still providing a one-to-one communication link. The second case enables one or multiple components to publish messages on queues which are then submitted to all components that subscribed to this queue.

Messaging infrastructure provides a number of quality of service guarantees that enable robust application design. Once the MOM acknowledges the hand-over of a message from the sending component, it guarantees delivery of this message to the intended recipient or alternatively a notification of the sender of a delivery failure message. A persistent, transactional storage (e.g. a database system) provides the basic capabilities for this. Similarly, for message delivery, the MOM guarantees delivery of a message exactly once (and redelivery in case of a receiver failure) until the receiving component acknowledges correct handling of the message. Certain messaging infrastructures enable a configuration of further quality-of-service parameters, priority delivery levels etc.

Java supports messaging through the JMS (Java Message Service) API, which enables access to any compliant messaging service provider. JMS is part of the Java EE capabilities.

The Advanced Message Queuing Protocol (AMQP) is an attempt to define behavior of the messaging server and client such that implementations are truly interoperable. The protocol provides point-to-point and publish/subscribe messaging, or combinations of both.

2.1.2 Service-Orientation

A second fundamental paradigm that is applied for the OOI CyberInfrastructure is service-orientation. Services are functionalities that are published on the network on known locations (addresses or inside known registries) and can be accessed by message exchange. Service-oriented architectures (SOA) are loosely coupled systems that focus on the provisioning of individual versatile, extendible, reusable services that contribute to the business goals of the system.

The loosely coupled Service-Oriented Architecture (SOA) is the key for the architecture which allows specific policies to be enforced by pluggable entities. SOAs have emerged as a widely accepted solution to this challenge; they use standards-based infrastructure to forge large-scale systems out of loosely coupled, interoperable services. SOAs can create systems-of-systems by mapping existing systems into services, then orchestrating communication between the services. To support continuous use, new functionality can be created by either adding new services or modifying communication among existing services.

Because of these features, SOA projects are particularly amenable to agile development processes involving extremely short development and redevelopment cycles. Consequently, well-executed SOAs can drive down financial and technical risk by improving flexibility and reducing time to market. As the number of stakeholders of a SOA project grows, so typically does the number and complexity of various business concerns (e.g., governance, security, and policy) and their level of distribution across the architecture. In order to maintain SOA's advantages, the integration of these concerns requires a framework that is scalable to complex systems and provides decoupling between the various concerns.

Web services provide a lightweight, well-established deployment pattern for capabilities in distributed systems. They are particularly attractive as a deployment technology because 1) they support a seamless transition from the service-oriented conceptual architecture to service-oriented deployment architecture, and 2) data and computing Grid technologies come equipped with web services interfaces, as exemplified by the ATOMIC interface of BIRN/Telescience to rationalize access to the national Grid Computing infrastructure. This results in low integration overhead.

2.1.3 Enterprise Service Bus

Enterprise Service Bus (ESB) technologies are rapidly emerging as the standard approach to system-of-systems integration. ESBs provide a highly scalable integration platform built on web and other open standards that combine access to heterogeneous data sources with messaging and web-service technologies to produce coherent, event-driven, service-oriented architectures that can rest on practically any transport and network access protocol. In essence, an ESB consists of four major components: a service/data interface, a messaging component, a router/interceptor component, and a set of infrastructure plug-ins.

The service/data interface acts as a gateway and adapter, connecting the ESB with other services and heterogeneous data and message sources, but also offering access to the services provided by components connected directly to the ESB. The service/data interface also serves as a gateway to other ESBs; enabling a wide range of integration topologies. The messaging component provides reliable messaging via freely configurable point-to-point and publish-subscribe communication channels.

The router/interceptor component captures messages intended for the messaging component and subjects them to a suite of infrastructure plug-ins according to a freely configurable routing scheme. Infrastructure plug-ins process messages they receive via the router/interceptor. Examples of plug-ins are data transformers, encryption engines, authentication, policy enactment, and failure management components. This combination of router/interceptor mechanism and infrastructure plug-ins is known as a dependency injection mechanism or aspect-oriented infrastructure. The message resulting from the routing/processing combination is then made available for consumption via the appropriate channel of the messaging component.

Further core ESB capabilities are:

- Message Transformation
- Message Enhancement
- Protocol Transformation
- Process Choreography
- Service Orchestration
- Transaction Management
- Security

Different ESB implementations provide different subsets of these core services plus further extended product specific services.

The main benefit of an ESB solution is that it enables the enactment of domain processes based on domain service descriptions that is isolated from technical concerns such as service implementation technologies, service orchestration, messaging technologies and message formats etc. The ESB provides the middleware to isolate the domain layer from the technical layer through a configurable mapping.

2.2 Rich Services Pattern and Architecture

The Rich Services Pattern [Arrott et al 2007] provides a logical system design pattern and deployment strategy for service-oriented systems, for instance implemented by ESB middleware platforms. The Rich Services Pattern is particularly suitable for systems of systems integration efforts.

The Rich-Service Architecture is a type of SOA that goes beyond by addressing the scalability of complex systems and provides decoupling between the various stakeholders' concerns, while enabling a direct and easy deployment mapping to runtime systems such as Enterprise Service Buses (ESBs), legacy platforms, Web Services, and combinations thereof. In this sense, the Rich Service architecture anticipates future trends and organizes systems-of-systems into a hierarchically decomposed structure that supports both "horizontal" and "vertical" service integration. Horizontal service integration refers to managing the interplay of application services and the corresponding crosscutting concerns at the same logical or deployment level. Vertical service integration refers to the hierarchical decomposition of one applica-

tion service (and the crosscutting concerns pertaining to this service) into a set of sub-services such that their environment is shielded from the structural and behavioral complexity of the embedded sub-services and the form of their composition.

This architecture is inspired by ESB architecture/ implementations, such as Mule and Cape Clear. Figure 1 illustrates the main entities of the architecture, which are (a) the Service/Data Connector, which serves as the sole mechanism for interaction between the Rich Service and its environment, (b) the Messenger and the Router/Interceptor, which together form the communication infrastructure, and (c) the plug-ins connected to Messenger and Router/Interceptor, which encapsulate various application and infrastructure functionalities. To address the horizontal integration challenge, the logical architecture is organized around a message-based communication infrastructure having two main layers. The Messenger layer is responsible for message transmission between endpoints. By providing the means for asynchronous messaging, the Messenger supports decoupling of Rich Services. The second layer, the Router/Interceptor, is in charge of intercepting messages placed on the Messenger, then routing them. The routing policies of the communication infrastructure are the heart of the Router/Interceptor layer. Leveraging the interceptor pattern readily facilitates dynamic behavior injection based on the interactions among Rich Services. This is useful for the injection of policies governing the integration of a set of horizontally decomposed services.

The main entity of the architecture is the notion of Rich Service. A Rich Service could be a simple functionality block such as a legacy system or Web service, or it could be hierarchically decomposed. There are two kinds of Rich Service Plug-ins: Rich Application Services (RAS) and Rich Infrastructure Services (RIS). RAS provide core application functionality, they mandate some communication to be initiated by some party in the business workflow and how the interaction is to be carried out. RIS address crosscutting concerns and do not initiate any communication by themselves. RIS can modify interactions defined by RAS by rerouting, filtering, or modifying in other ways the messages exchanged.

Policy enforcement, encryption, failure management, and authentication are typical examples of RIS, which are mainly intermediary services. The purpose of an encryption service, for instance, is to ensure that all messages transmitted over the communication medium are encrypted. A traditional service approach would require modifications to every service in the system in order to integrate the encryption mechanism, leading to scattered functionality. On the other hand, the Rich Service architecture introduces encryption as an intermediary service that can inform the Router/Interceptor layer to modify the message routing tables to ensure that every message sent to the external network must first be processed by the

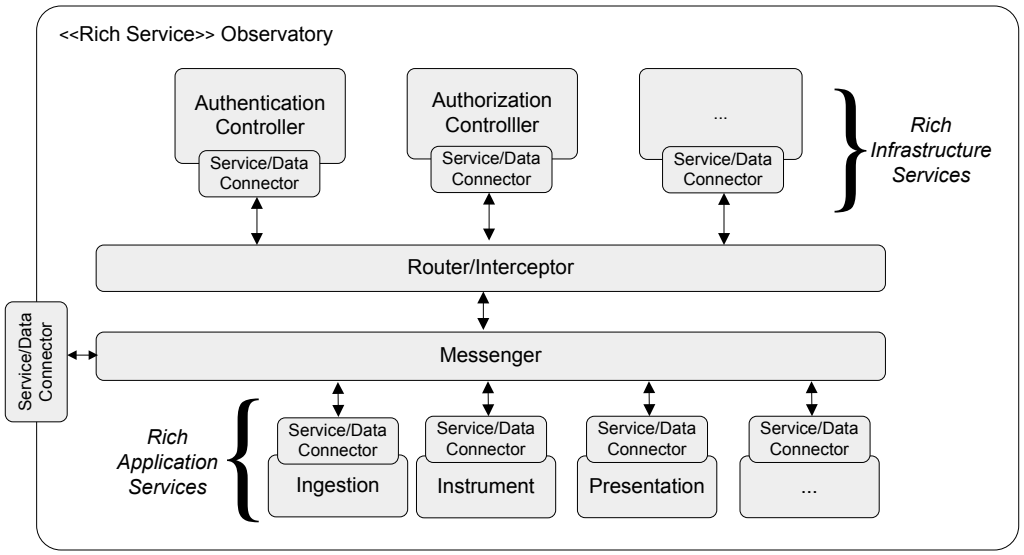

Figure 1. Rich Service logical deployment pattern

encryption service. Only the communication infrastructure needs to be aware of the encryption service. As such, the architecture lends itself to both modern and legacy equipment, not constrained to any single platform, product or vendor.

A Service/Data Connector is the means by which Rich Services are connected to the communication infrastructure. It encapsulates and hides the internal structure of the connected Rich Service, and exports only the description and interfaces that the connected Rich Service intends to provide and make visible externally. The communication infrastructure is only aware of the Service/Data Connector, and does not need to know any other information about the internal structure of the Rich Service. This helps tackle the vertical integration challenge introduced by systems-of-systems. The interface of a Rich Service has associated a structural and a behavioral view.

In this architecture, each Rich Service can be decomposed into further Rich Services, with the Service/Data Connector acting as a gateway responsible for routing messages between layers. This is illustrated in Figure 2. Additionally, the use of the Router/Interceptor layer removes dependencies between services and their relative locations in the logical hierarchy.

This approach enables services from different levels of a hierarchy, possibly with different properties (such as encryption and security requirements) to interact with each other seamlessly without ever being aware of such incompatibilities. In addition, this architecture can support the specification of dynamic systems, where new services can be introduced at runtime and provide their functionalities without the need to change or adapt the implementation of existing services. Any required adaptation or rerouting can be handled by adding application or infrastructure services to the system. Service composition can be addressed in the system by following two general approaches. One is to add a new RAS, acting as an orchestrator, which utilizes the existing services to provide the composite service. Another approach is to add a RIS that intercepts messages coming from/going to a specific RAS.

Another important benefit of the Rich Services architecture is the ability to analyze, design and decompose the system according to various major concerns (e.g., security, resource constraints, policies, and governance) in isolation. Each major concern can be designed separately, and then made available to the entire Rich Service via the Router/Interceptor layer. This separate design of common concerns results in reduced implicit coupling among the Rich Service's constituent services. This feature greatly contributes to the design of Very Large Scale systems where multiple authorities are involved.

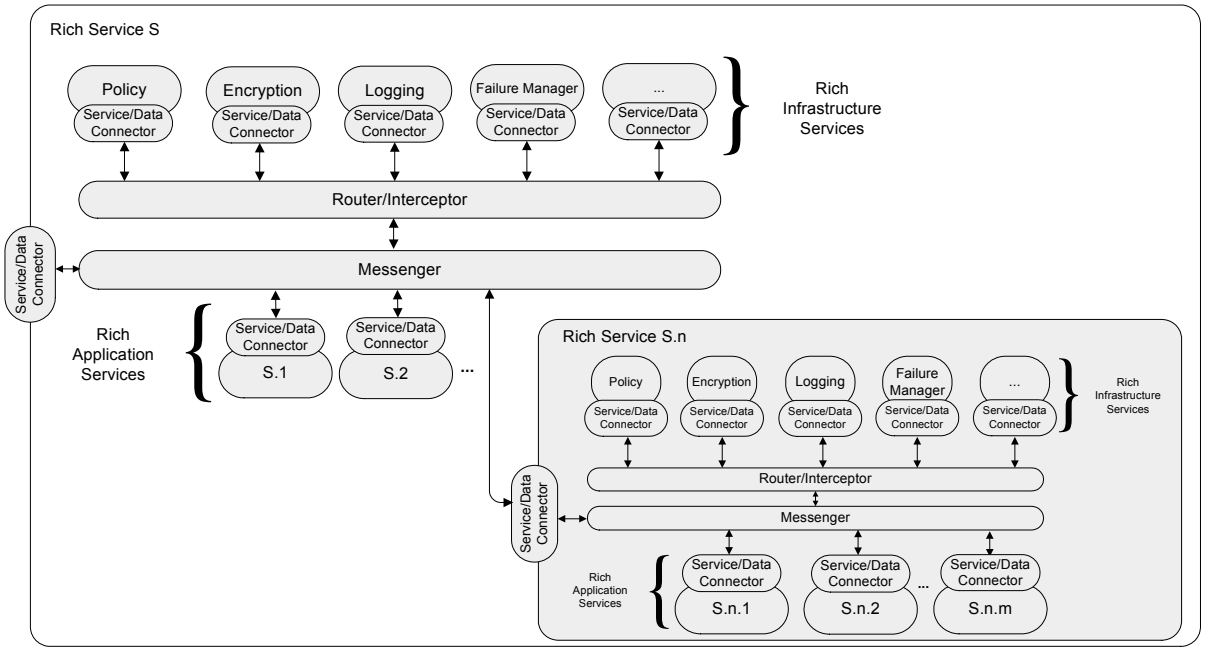

Figure 2. A hierarchical logical architecture – the Composite Rich Services

As an example, from the security viewpoint, multiple security-related services can be designed and connected to the system at different levels in the Rich Service hierarchy to address the authentication, access control permissions, and privileges across multiple authority domains. Self-monitoring and reporting can also be easily integrated as RIS to report on message throughput, application performance, or any other relevant metrics.

Rich Services can easily map to an implementation architecture based on an ESB solution, a threaded execution solution, or a range of other solutions. ESBs, in particular, provide message routing amongst well-defined, loosely coupled, coarsely granular services similar to our notion of Rich Services. They achieve cost efficiency and flexibility through the integration and reuse of services and service-based applications. Additionally, ESBs may employ many routing strategies (e.g., static routing, content-based rules, and itinerary) to achieve architectural goals such as distributed execution, load balancing, flow throttling, auditing, redundancy, failover, online maintenance, provisioning, and minimal service interdependence. This makes them particularly attractive as deployment architectures. The Rich Service architecture can be leveraged into an end-to-end software engineering process that ranges from requirements gathering to physical network deployment and fits well system-of-systems integration processes.

Most importantly, the scale-independent logical structure Rich Service Architecture supports a very broad suite of capabilities and will allow users well into the future to compose capabilities that are not now obvious. This abstraction is critical to future success and acceptance of the OOI CI.

2.3 The CI Capability Container

2.3.1 Provided Capabilities and Services

The CI Capability Container provides a rich infrastructure environment for applications and services. It discharges a most infrastructure service needs that are required for any OOI and CI application services. Capability Container is based on the Rich Service Architecture blueprint.

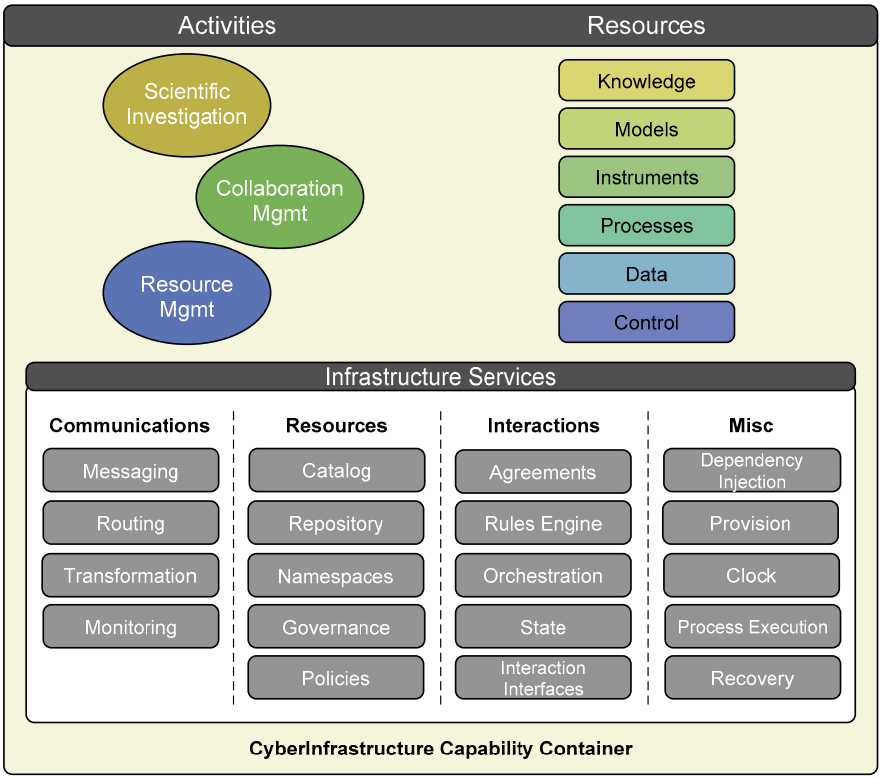

Figure 3. CI Capability Container Environment

Figure 3 shows a schematic overview graphic of a Capability Containers and its rich set of infrastructure services as listed above. The pervasive provisioning of these infrastructure services across the deployment architecture constitutes the core of the Common Operating Infrastructure (COI). A Capability Container provides access to the resource networks via service interfaces. Each Capability Container also has a presentation capability to project its services to its environment in various forms.

The Capability Container provides the following infrastructure capabilities:

Capability	Description
Messaging	Receiving and delivering messages on queues
Message routing	Storing and forwarding messages to their destinations according to configured routes
Message transformation	Changing message format and content while in transit to mediate flexibly between message sender and receiver
Monitoring	Observing the flow of messages and providing statistics
Catalog	Keeping track of all the resources and enabling discovery, querying
Repository	Storing and retrieving data elements
Namespaces	Providing categorization and isolation for resources of different topics
Governance	Keeping track of the life cycle of resources and multi-party interactions accessing the resources
Policy enforcement and update	Defining how resources can be accessed depending on the owner, operator and accessing party. Enforcing the application of the policy and providing means to compose policies from different sources and update them dynamically throughout the system
Agreement management	Negotiating agreements and contracts between federated parties; keeping track of commitments and obligations; enforcing their enactment
Conversation orchestration	Coordination of interactions among distributed parties in an organized way towards reaching I defined goal
State	Keeping track of the stateful information or session related to a resource or an interaction conversation.
Interaction Interfaces	Defining and enacting templates for interactions and protocols as sequences of information/message exchange as means for interface definition
Provisioning	Providing or executing services on behalf of other users in the system
Clock	Providing a synchronized common clock throughout the system that is accessible by all applications and services and which provides sufficient precision
Recovery	Robust handling of exceptional system states and the ability to recreate a recent known consistent state throughout a conversation or the system
Process execution and computation	Providing dynamic computation capability to various users of the system in a defined and controlled way.

Not all capability containers need to provide all infrastructure services, depending on the deployment needs. SV-1 identifies a number of deployment configurations for CyberInfrastructure Points-of-Presence, which determine the set of capabilities required for each of these configurations.

Because of their pervasive nature, the Capability Containers are ideally suited to addressing cross-cutting infrastructure concerns, including security, reliability, governance, and scalability. Capability Containers enable an easy deployment of the OOI collaboration and policy framework.

2.3.2 Capability Blocks

These ESB characteristics are foundational for the implementation of capability blocks, as shown in Figure 4 (Composite ESB Design Pattern). A capability block can be either simple or composite. In this figure, there are capability blocks on the highest level for instance for observatories, modeling facilities and research laboratories. These capability blocks all have interfaces to the Common Operating Infrastructure (COI), itself a capability block.

Capability blocks on this high level may for instance represent domains organizational authority. The capability blocks for the regional observatory in this picture shows another level of nesting. Inside are further capability blocks connected to another instance of the COI. On this level, for instance, Capability

Containers as introduces above can realize the capability blocks. The observatory node block shows a third level of nesting with capability blocks and a COI instance inside. This composite pattern is universal and can occur at any level in the design of the system.

An example of a simple capability block is an instrument proxy, as shown by the Interface Point in Figure 5 (Example Capability Container Deployment Model); it displays no further decomposition, but rather exposes its capability as a web service. A composite capability block is comprised of the ESB parts mentioned above, bundled with a specific set of plug-ins and additional hierarchically-composed capability blocks that are connected, via their service/data interfaces, to the messaging component of the composite capability block.

An example of a composite capability block is the overall OOI CI. Most importantly, the scale-independent logical structure supports a very broad suite of capabilities and will allow users well into the future to compose capabilities that are not now obvious. This abstraction is critical to future success and acceptance of the OOI CI.

At each deployment site, a Cap Container implements the capability block pattern as described above. In particular, the infrastructure services will be provided as ESB-plug-ins and the resource networks will be integrated via web services, or as capability blocks in their own right. OOI activities emerge from the interplay of multiple capability blocks, as shown by the data collection activity example.

2.3.3 Example Capability Container Deployment Scenario

Figure 5 (Example Capability Container Deployment Model) illustrates the layout of a Cap Container for the data collection activity scenario. Because of its resource-constrained nature, the Instrument Point Cap Container provides only general CI infrastructure and instrument proxy services. The Acquisition Point Cap Container illustrates the use of infrastructure services to implement the processes in the data collection activity scenario. The process execution infrastructure service provides the filtering and triggering processes at this Cap Container. An additional function supported by the Cap Container is the presenta-

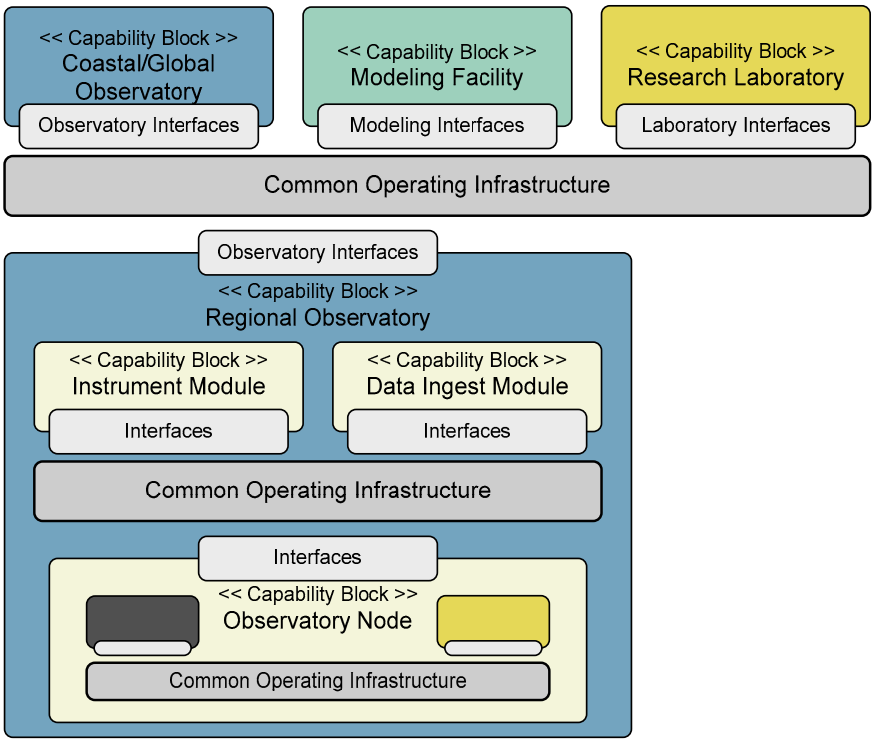

Figure 4. Composite ESB Design Pattern

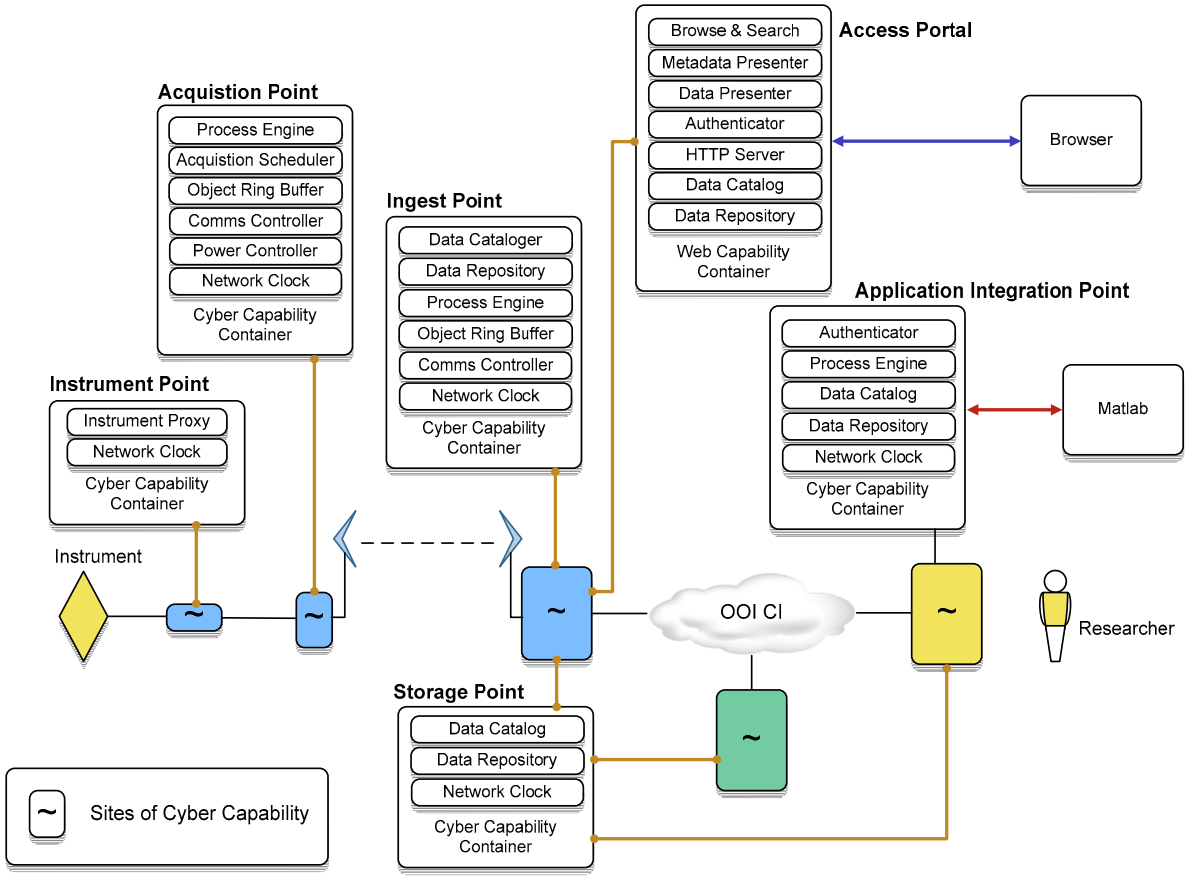

Figure 5. Example Capability Container Deployment Model

tion capability implemented in the Access Portal Cap Container. It will contain portlets for session management and Google Earth presentation built on the BIRN/Telescience GridSphere-based portal framework, as well as an http container.

We describe the infrastructure elements that support the Data Collection Activity of a global scale, buoyed observatory as one relevant deployment scenario. This spans the entire range of deployed systems and networks from ocean-based instruments to CyberPoPs and user applications.

The Instrument Point represents the interface between the physical world and the CI; it comprises proxies that provide a programming interface to the instruments. The Acquisition Point provides instrument control and data acquisition and transmission functions. It comprises a process and instrument controller and a data acquisition subsystem. Researcher-supplied triggers initiate data acquisition processes that the process controller translates into commands for the instrument controller.

Data from the instruments transit from the instrument controller to the acquisition component, where researcher-supplied filters result in either new data acquisition or transmission of the data to the Ingest Point. Data are sectioned appropriately by a segmentation component for transmission and handed over to the transport broker, which uses the Object Ring Buffer (ORB) and the communications controller to locally store and transmit the data, respectively, based on network availability and Quality-of-Service (QoS) constraints. At the Ingest Point, data arrive via the local communications controller and transport broker. The latter feeds data correction and ingestion components. The ingested data, along with their metadata, are buffered via the local storage broker.

The storage broker interacts with the Storage Point that offers repositories for data and services, as well as data and metadata cataloging. The researcher has multiple ways to view and further process data. First, an Access Portal supports data access and presentation via web- and other browsers. It interfaces

with the Storage Point via a local storage broker and offers components for search, navigation, and presentation. Second, an Application Integration Point supports data access, transformation and analysis via a set of analysis tools. It also connects to the Storage Point via a local storage broker, and offers programming interfaces so that researcher-supplied analysis and transformation processes can access and manipulate the data.

In this deployment scenario, the Instrument Network comprising the Instrument Point, the process and instrument controller, acquisition and segmentation components, transport broker, ORB and communications controller of the Acquisition Point, as well as the communications controller and transport broker of the Ingest Point are built from BRTT Antelope® components. The Data Network comprising the storage provider with its repositories and catalogs at the Storage Point and the federated storage brokers of the Ingest Point, Access Portal, and Application Integration Point are built from UCSD Storage Resource Broker (SRB) components. Presently, v1.0 of the ROADNet PoP (RPoP) integrates Antelope and SRB in a small, low-power, low-cost LINUX box using Intel XScale Processors. The next operational release will include web services support.

The Processing Network is implemented using researcher-provided filter and trigger processes in the Acquisition Point, a data correction process in the Ingest Point, a presentation process in the Access Portal, and the transformation and analysis processes in the Application Integration Point, where the visualization and modeling capabilities of, for example MatLab, are also provisioned.

The data management functionality of the Data Network, comprising ingest and metadata cataloger components at the Ingest Point, the metadata-based search and navigate components of the Access Portal, and the navigate component of the Application Integration point, are implemented using components of the MBARI Shore Side Data System (SSDS). The repositories and catalogs at the Storage Point are implemented using OOI-specific adaptations of SRB repositories and catalogs currently deployed for BIRN/Telescience, with necessary extensions to house service repositories. All infrastructure elements are implemented using Rich Services Deployment Pattern. They are showed with their particular plug-ins.

2.3.4 CyberInfrastructure Point of Presence

The primary compute and storage building block for the CI is the CyberInfrastructure Point of Presence (CyberPoP). CyberPoP implements CI Capability Container described in the sections above. From the perspective of users, observatory operators, and the CI, CyberPoPs are virtual resources that can be provided in ways ranging from actual hardware installations to an elastic service analogous to Amazon®'s Elastic Compute Cloud®.

There are four classes of CyberPoP provisioned by the proposed work; the on-demand highly scalable (i.e., elastic) CI CyberPoP; the high availability, shore side, Observatory CyberPoP; the ocean deployable Marine CyberPoP; and the high bandwidth stream processor (HBSP) CyberPoP.

The highly scalable CI CyberPoPs will be accessible at two central facilities (one at SDSC based on Teragrid, and one whose location is yet to be determined). These two installations will serve as the central repository and core processing centers for OOI.

2.4 Conversation Management

2.4.1 Interaction interfaces

OOI activities will be enabled by the precise specification of the collaboration pattern, including the required subsystems, their interaction protocols, and a description of the information exchanges over time with each observatory activity, through an interaction interface. Furthermore, cross-cutting authentication, security, governance, and policy requirements will be associated with each interaction interface.

The interaction interfaces will be provisioned via the COI so they will be bound to actual resources either at the time of deployment or at runtime to provide the required degree of flexibility in system configuration. In effect, the OOI activity model is mapped to a service-oriented process model that is supported by appropriate configuration of the orchestration plug-in of each Cap Container.

2.4.2 Collaboration and Policy Framework

The CI Cap container provides capabilities for collaboration, agreement support and policy enforcement. Figure 6 illustrates this pattern for the base case of a single service provider and consumer. The pattern generalizes to arbitrary numbers of participants in a service orchestration. Conceptually, the example captures the establishment of a service agreement between two parties; for example, this could unfold between a regional cabled observatory (service provider) and a buoy-based global observatory (service consumer). Each one of the parties has established contractual commitments with their respective user communities, including membership agreements. Upon establishing mutual commitments, a contract between the two parties is in place. Furthermore, each party operates under a set of policies. The negotiation and contracting process, as well as the actual service usage, leads to an interaction pattern between the two parties that is constrained by the contractual commitments and policy subscriptions of both parties.

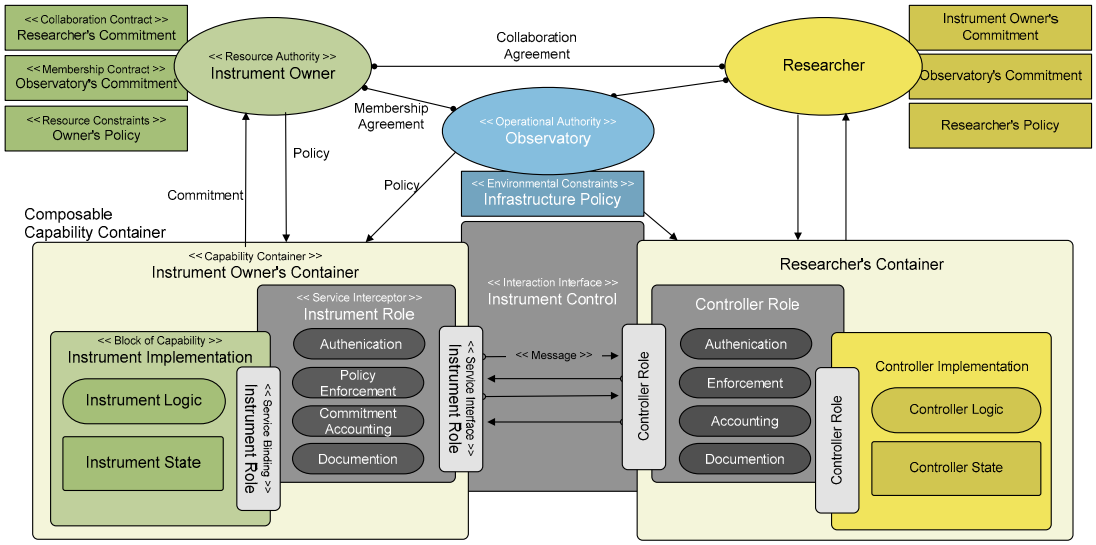

Figure 6. Collaboration and Policy Framework

Because each Cap Container is equipped with plug-ins for orchestration, governance, policy enforcement, and monitoring/audit, the deployment mapping for the collaboration and policy framework is straightforward: the corresponding interaction interface is stored and accessed COI-wide. Each party's Cap Container orchestration component executes the projection of the interaction pattern on the respective role to participate in the overall collaboration. The governance and policy constraints are extracted from the interaction interface and provided to the corresponding Cap Container plug-ins for monitoring and enforcement.

The Data Acquisition example provided above shows how the COI facilitates the interoperability and on demand mobility of the capabilities of the Instrument, Processing, Data, and Knowledge resource networks. The core software abstraction illustrates how the COI, through the use of the CI capability container, factors out the common aspects of communication, state management, execution, governance, and service presentation to provide a very scalable, secure and extensible model for managing user-defined collections of information and taskable resources. This ability to integrated resources of different types implemented by different technologies is the central proposition of the architecture. It provides the basis for an integrated observatory network that will remain viable and pertinent over multiple decades.

3 Identification of System Nodes

The OOI architecture rests on a rigorous service-oriented design approach to provide subsystem capabilities and their integration into system capabilities. This yields a seamless software and system engineering framework. Intuitively, every OOI entity (for instance instruments; laboratories; data repositories; coastal, regional, and global observatories; the computational grid; observatory management) will represent itself as a set of services that can be located and accessed via the CyberInfrastructure. Web services and related technologies will enable rapid implementation, provisioning, and integration, along with flexible configuration of these services to yield the CI.
The resource network concept facilitates the federation, governance, and management of interactions between system activities and a base set of resource types. By leveraging and structuring the shared characteristics of the complete set of resource elements, the architecture provides an extensible model for the operation, presentation, and composition of resources. The architecture then provides a standard basis for provisioning, managing, and sharing resources across the physical, technical, and organizational domains of the observatory network. This resource network model provides a consistent and scalable approach to the federation, governance, and operation of all types of resources; one that enables pervasive and universal access without compromising control and security.

The CI consists of six services networks that contain different classes of resources and the capabilities needed to manage them. These resources are tied together by two cross-cutting infrastructure elements:

1. The Common Operating Infrastructure (COI) serves as the integration platform and communication conduit, and supports the activity, resource, service, identity, communication, and presentation models that must be used across OOI.
2. The Common Execution Infrastructure (CEI) is the computational substrate that supports provisioning, configuration and direction of collection and analysis protocols.

Resource networks as well as infrastructure elements realize the highest level architectural elements of the CyberInfrastructure.

3.1 Common Operating Infrastructure (COI)

The COI provides the technologies and services to play the role of 1) a unifying information conduit, enabling data and control streams to be published and consumed by all of the subsystems; 2) a platform to execute the core elements of the activity model by allowing the subsystem services to be combined as workflows; and 3) the implementation location for cross-cutting aspects of the CyberInfrastructure.

3.1.1 Capabilities

The COI should provide the following capabilities:

1. Collaboration provisioning and agreement management,
2. facility provisioning and rights management,
3. identity validation and verification,
4. Service provisioning and interchange management,
5. federation and delegation of service presentation and fulfillment,
6. resource collection management, navigation and search,
7. resource lifecycle management,
8. policy enactment and enforcement management, and
9. communication provisioning and interchange management.

3.1.2 Technologies & Systems

The COI will be realized as a composite capability block providing 1) an integration platform for data and control channels (streams), block data transfer and streaming media, 2) a rich set of options for integrating heterogeneous data sources and applications using a variety of data transports, and 3) an interface for injecting infrastructure services, such as policy monitoring and enforcement services, as plug-ins to effect federated authentication and security policies. This is the basis for provisioning the service (registry, brokering, binding and execution of services), facility and resource networks within the COI.

The services and their corresponding data models provide a uniform mechanism to detect and exploit the capabilities of OOI entities. Governance will be supported through templates for collaboration agreements that define partnership, and delegation and federation policies for participants and their assets. In particular, we will establish an integrated policy/security framework that is directly coupled to the interaction interfaces capturing the activity models.

Design partners NCSA and NCSU have developed a comprehensive architectural framework for policy and security management that ties in directly with the notion of interaction interface. Policy and security properties are stored together with the interaction protocol that defines a service using the zone federation architecture of the SRB. This framework will be injected into the capability block pattern such that all interactions with the COI directly fall under the governance of this policy framework. Specifically, the ESB dependency injection mechanism and GridShib, GridGrouper, and myVocs integrated with SRB's Zones will be used as the core technology for implementing and enforcing the COI data structures and models for governance, security and other policies.

In the initial deployment, the capability blocks will be populated as follows: the messaging component will be instantiated to ActiveMQ/AMQP and the router/interceptor and service/data interfaces will be instantiated to Mule. Mule's dependency injection mechanism provides immediate access to persistence, application/transaction, workflow, configuration and monitoring frameworks that will be instantiated to Hibernate, Spring, Groovy and JMX, respectively. Furthermore, the COI will leverage the successful CI software stacks of BIRN/TeleScience with their web-service-based ATOMIC interfaces to the national Grid computing and security infrastructure. All of these web services will be configured as capability block plug-ins. The flexibility of this ESB-based approach allows the development team to rapidly integrate new capabilities as they become available.

The transport-transparent messaging component of the ESB will be exploited to implement data and control streams among CI subsystems, and provision and broker any service, data source, data transport and delivery mechanism, as well as any policy that is injected into the routing/interceptor facility of the ESB. Such a messaging system supports secure, durable, fault tolerant and high availability connections.

3.1.3 Data Structures & Models

The architecture encompasses authoring, implementing and maintaining the key conceptual model underlying the patterns of governance that reflect security considerations as well as the OOI activity model. These patterns will be stored and made accessible via a Data Network repository for high-quality governance management.

3.1.4 Interface Points

The COI is the communication and integration substrate for the system; it, therefore, interfaces with all subsystems, as well as with the environment via the data/service interface of the ESB.

3.1.5 Systems and Interfaces Assignment

Table 1 depicts the assignment of systems and interfaces to operational nodes and needlines from OV-2 in COI diagram. Figure 6 shows these assignments from OV-2 for COI (systems are highlighted with yellow color, operational nodes with white color). Further, operational node Data Network will use services from systems Database, SRB Zones and ROADNet. Operational node Instrument Network will use services

from systems SIAM/SSDS. All interfaces between Mule and the rest of the systems are realized using ActiveMQ

Operational Node/Needline OV-2	System/Interface SV-1
Router/Interceptor O6 Messenger/Communicator O7	Mule
Identity Management O1	NCSA Identity Management
Authentication O2	NCSA Authentication
Policy Enforcement O3	NCSU Policy Enforcement
ID Information ON1 Authentication ON2 Policy ON3 Event ON4	Active MQ

Table 1. Common Operating Infrastructure – System and Interfaces assignment

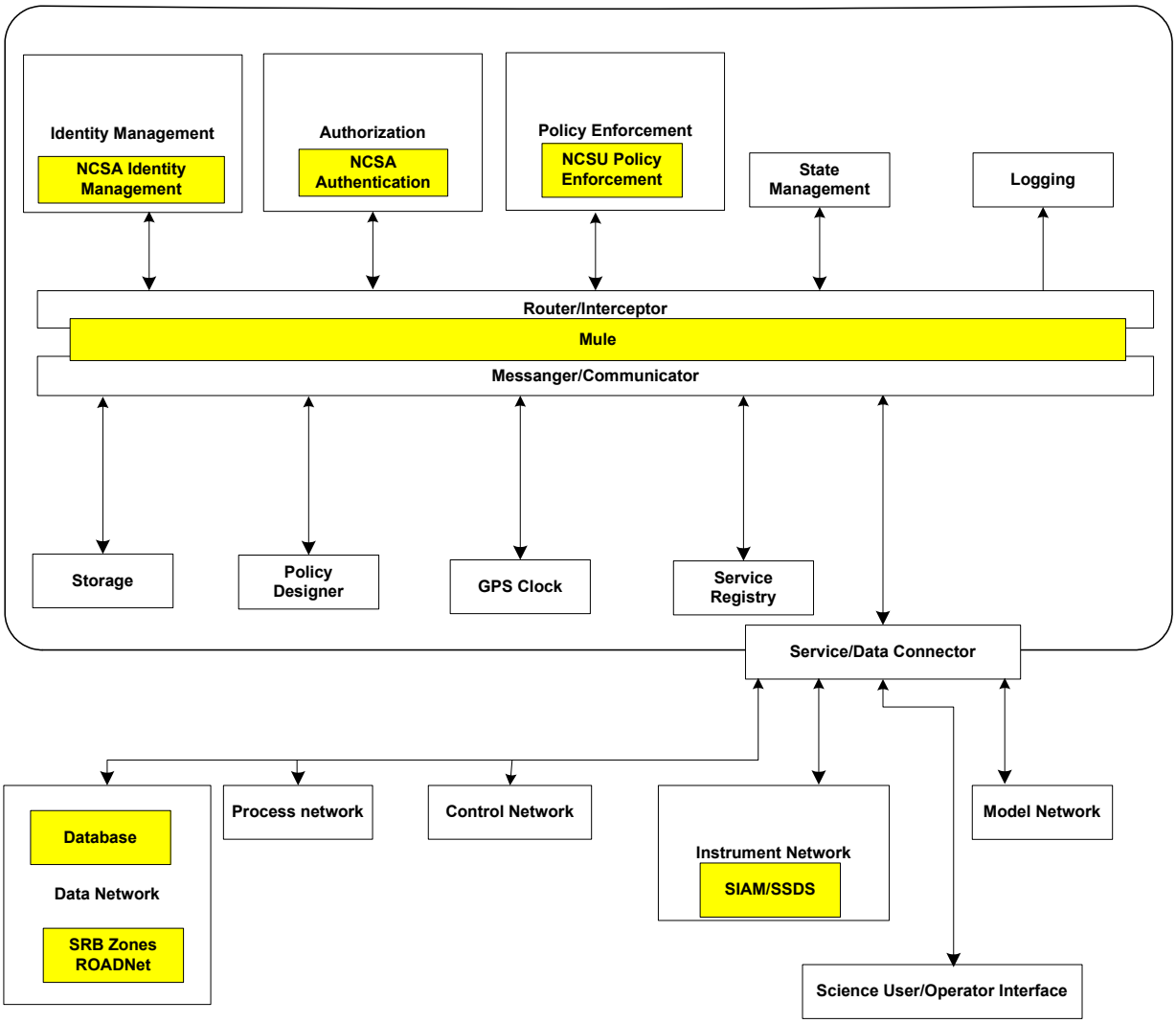

Figure 7. Common Operating Infrastructure - System and Interfaces assignment

3.2 Common Execution Infrastructure (CEI)

The CEI will provision the services required to implement an elastic compute network together with a corresponding management UI module. This constitutes the computation and execution substrate for the entire CI.

3.2.1 Capabilities

The CEI should provide the following capabilities:

1. Virtualized computing resource provisioning, operations and maintenance,
2. provision parameterized configurations of service and application modules into compute node deployment packages,
3. monitor and provision compute nodes based on compute resource utilization and latency of provisioning,
4. on-demand scheduling of processes,
5. optimized scheduling of stream process subscriptions,
6. extendable process execution environment that supports multiple execution formats,
7. federation of process execution service providers, and
8. process control interface.

3.2.2 Technologies & Systems

The technical components of the CEI include a virtual compute node configuration repository, a virtual compute node loader, a virtual compute node deployment scheduler, and a base virtual compute node; the implementation will leverage Globus' Virtual Workspaces. Furthermore, the CEI will provide virtual compute node test and certification services to support its monitoring and management using an adaptation of the ROCKS roll retrieval and system configuration mechanisms. A system for administering virtual compute node configuration repositories and retrieving both the contents and their catalog will be defined based on the Xen technology, and connected with the Data Network. The virtual compute node loader will retrieve elements from one or more local or system repositories and use them to configure a virtual compute node.

3.2.3 Data Structures & Models

Globus' Virtual Workspaces and Amazon's Elastic Compute Cloud (EC2) provide the base models for the Common Execution Infrastructure.

3.2.4 Interface Points

The CEI provides its services via the COI to all CI subsystems. However, major interaction points exist between the COI and the Control, Processing, Modeling and Data Networks.

3.3 Control Network (CN)

The Control Network will provision the services required to establish the standard models for the management of stateful and taskable resources. It provides controller processes the semantics to monitor and control the operating state of an active resource as well as to initiate, monitor and amend tasks being carried out by a taskable resource. The managed resource is required to present their operational state and declare their governance context.

3.3.1 Capabilities

The CN should provide the following capabities:

1. Provisioning the command, control and monitor semantics to operate and manage a resource,
2. Time structured concurrent coordination and prioritization of shared resources that are distributed and constrained,

3. Provisioning of a behavior-based architecture for rapidly reconfigurable autonomous task reconfiguration,
4. Unique multi-objective optimization of behavior coordination, allowing for effective compromise to be attained between periodically competing task objectives for a collection of resources,
5. Provisioning of a behavior calculus, allowing sequences of task states to be structured for long-term, persistent plans while remaining highly reactive to events and in-situ control requests.

3.3.2 Technologies & Systems

The deployment of the technical components of the Control Network is predicated on the ESB implementation that is the basis of the capability container concept of the COI–this allows the control network to have a federated presence across the CI. In particular, the management of state, execution scheduling and orchestration of taskable resources will be provisioned as state management and orchestration/process execution plug-ins of the ESB. Further techniques and implementation technologies on which the Control Network is predicated include: Interval programming (IvP), a unique, new mathematical model for representing and solving multi-objective optimization problems for reconciling vehicle behaviors active during a mission; MOOS (Mission Oriented Operating Suite), an open source middleware for connecting software components on an autonomous platform; IvP Helm, a behavior-based autonomy package using multi-objective optimization for behavior reconciliation, with a full Boolean logic behavior calculus and an interface to the MOOS middleware.

3.3.3 Data Structures & Models

The taskable or executive resource is at the center of a set of resources comprised of the Task, Plan, Scheduler and Executive. The Task is a specification that declares the execution statement as well as its pre- and post-conditions. An execution statement is either a basic command (such as a Data Network query) or a composite (such as a batch script or binary executable). The Plan resource couples Tasks to time and/or information events using Event-Condition-Action rules. The Scheduler coordinates the capacity, and constraints of the execution environment with the demand and priority of tasks. The Executive resource provides the controlling process with the ability to assess, prosecute and monitor a Task.

3.3.4 Interface Points

The Control Network is used as the basis for and thus has major interaction points with: the Processing Network, the Instrument Network, the Modeling Network; the provisioning model for the CEI's Elastic Compute Cloud.

3.4 Data Network (DN)

The Data Network provides services for the secure preservation, management and presentation of scientific data associated with their structural and semantic context. It supports enactment and enforcement of the OOI Data Policy. The repository and its rule-based policy system will enable and support system-wide registration, persistence and presentation of resources. Each resource is mapped onto a logical name space, and persistent attributes are then managed for each object within that logical name space. Finally, the Data Network will integrate with digital library technologies and preservation environments for publishing and archiving data streams and derived data products.

3.4.1 Capabilities

The DN should provide the following capabilities:

1. Provision, manage and present data and metadata supporting the OOI domain and data models,
2. provide policy-governed data access,
3. provide user-defined data presentation,
4. provision, manage and present data repositories, collections and streams,

5. negotiate and manage federations of data repositories, collections and streams,
6. negotiate and manage delegation of data preservation and presentation responsibilities, and
7. maintain and ensure the integrity of data in perpetuity.

3.4.2 Technologies & Systems

The core element of the Data Network is the Storage Resource Broker (SRB) data Grid, which is in production use within the ROADNet project to manage federation of multiple independent data streams. SRB data Grids have also been integrated with digital library and knowledge management technology (DSpace and Fedora), and have been used to build preservation environments. Furthermore, the SRB is in production use on the BIRN/Telescience projects, where it acts as a federated data store for both metadata catalogs and vast volumes of imaging and other data. The integrated Rule-Oriented Data System (iRODS) adds policy-based data management capability to the SRB environment. MBARI's Shore Side Data System (SSDS) will be integrated with SRB/iRODS to provide domain-specific data cataloging, search and navigation capabilities.

3.4.3 Data Structures & Models

The project will leverage and adapt the successful SRB implementations, featuring data, catalog and repository models for ROADNet and Telescience/BIRN, integrating IRIS's Standard for the Exchange of Earthquake Data (SEED), and MBARI's SIAM/SSDS.

3.4.4 Interface Points

The major interaction points of the Data Network are with the Instrument Network through receipt of data streams packetized into files that are stored on repository systems specified by management policies. The Data Network interacts with the CEI by submission of long-running tasks such as checksum-based integrity validation on a collection to a workflow environment for execution. The DN also interacts with the COI to observe the OOI policies managed within that element, as well as to provide data access for presentation and transformation purposes. External data access services will be provisioned leveraging UCAR's THREDDS, OPeNDAP & IDD, as well as OGC Web.

3.5 Processing Network (PN)

The Processing Network provides immediate-mode scheduling of processes at specified locations based on explicit time requirements and event triggers. The processes to be scheduled and executed come in a variety of forms, including compiled code, scripting languages, and workflow graphs, and are placed within the common time and event semantics of the COI.

3.5.1 Capabilities

The PN should provide the following capabilities:

1. Immediate-mode scheduling of processes at specified locations,
2. coupling of processes to the streaming environment of the Data Network,
3. coordinated and/or chained scheduling of processes,
4. an extendable set of process execution engines,
5. standard process execution planning and control, and
6. standard providence capture and reporting. Process authoring and monitoring applications that will be integrated as a user interface to the Processing Network include MatLab and Kepler, as well as NCSA's Community Ensemble Service.

3.5.2 Technologies & Systems

The scheduler will be built based on NCSA's Community Ensemble Service. The core execution engines will leverage and adapt the existing Kepler framework, as well as the execution engines of BRTT Antelope, JJDAC, ISI Pegasus, and Matlab.

3.5.3 Data Structures & Models

The process and scheduling models will be based on the UCSD's Kepler scientific workflow system and ISI's Pegasus Grid workflow execution framework.

3.5.4 Interface Points

The major interaction point for the Processing Network is the streaming environment of the Data Network. The coupling of processes is primarily accomplished through publication and subscription to the Data Network. The COI, especially its CICCs, provision the execution infrastructure, as well as access to the CEI elastic node provisioning service and the Knowledge Network resource association and tagging services.

3.6 Instrument Network (IN)

The Instrument Network will implement services falling into the categories of network-wide sensor Grid configuration and control, system support capabilities for individual instruments on the Grid, and support for various aspects of the entire mission (i.e., domain models). The Instrument Network will provide facility components, such as instrument management and presentation modules, as well as services for taskable interfacing with instruments, and instrument and metadata registration.

3.6.1 Capabilities

The IN should provide the following capabilities:

1. Provide data acquisition, buffering, and transport mechanisms,
2. provide command and control systems,
3. maximize total data return from all instruments,
4. provide instrument test and certification,
5. provide instrument registration with associated metadata,
6. place processing capability into the instrument network,
7. manage and allocate resources to instruments,
8. manage and allocate resources from instruments,
9. acquire, marshal, buffer, and transport data to the Data Network,
10. prioritize data delivery, and
11. provision storage and processing throughout the Instrument Network.

3.6.2 Technologies & Systems

The Instrument Network will initially use the ORBserver technology as the backbone of the transport system. This is a reliable, content-neutral, packetized buffering and eventdriven distribution system that can be configured for gridded data acquisition, sharing, and processing. Extensive measures have been taken in its implementation to assure robust delivery. Network state-of-health monitoring issues have already been addressed in part by land-based sensor grids such as ROADNet, based in part on the Nagios open-source network monitoring package (http://www.nagios.com) together with
local modifications.

3.6.3 Data Structures & Models

The Instrument Network will implement and contribute domain models via the COI for instruments, observations, plans, schedules, marine resources, allocation, and transducers, leveraging standards such as

OGC Sensor Web Enablement (SWE) and IEEE 1451. The plug-and-play instrument support and remote control capabilities of the Software Infrastructure and Applications for Monterey Ocean Observing System SIAM will provide models from which the OOI CI will draw.

3.6.4 Interface Points

The Instrument Network provides its services via the COI to all subsystems. However, major interaction points exist with the COI, the Control Network and the Data Network. While the COI should largely mitigate end-chain communication technology limitations, it is clear that the pathways used for data acquisition and state-of-health monitoring for instruments must often share features with network-wide resource allocation, network state-of-health monitoring, and sometimes data Grid transmission capabilities. Thus, the Instrument Network must at the very least peacefully coexist with features of network-wide communication from central observatory acquisition and control sites out to those of sensors. The Instrument Network interacts heavily with the front-end of the Data Network, and hence must partially drive the interface engineering.

3.7 Modeling Network (MN)

The Modeling Network will provision ocean models and provide service implementations for their exploitation. To that end, it will incorporate 1) a virtual ocean model covering the full range of observatory scales; 2) a model control and virtual sampling interface; 3) data access and assimilation interface; 4) model coupling interface; 5) a taskable resource interface supporting construction, modification and execution of numerical ocean models, as well as Observing System Simulation Experiments (OSSEs) for network design and trade studies, data impact investigations, and data and information management exercises.

3.7.1 Capabilities

The MN should provide the following capabilities:

1. Data calibration and validation,
2. derived data product generation,
3. a stream process network, including stream subscription with process and execution location, stream process scheduling, and stream process execution,
4. a measurement processing network, including a measurement calculus and measurement semantic model, and
5. a modeling network, including on-demand modeling, assimilative modeling, and an observing system simulator.

3.7.2 Technologies & Systems

The development effort for the Modeling Network will rest on community-based numerical ocean models such as the Regional Ocean Modeling System (ROMS) and the Harvard Ocean Prediction System (HOPS). Its modeling and simulation capabilities will leverage existing and emerging data assimilation modules based on the variational method (e.g., 3DVAR or 4DVAR) or Kalman Filter (KF). The MN will be configured as a set of web services with a portlet implementation for model configuration, data delivery and visualization.

3.7.3 Data Structures & Models

JPL's OurOcean Portal, ROMS and HOPS will provide the core data models for the Modeling Network.

3.7.4 Interface Points

The Modeling Network will interface with the COI to interact with the other resource networks and infrastructure elements. In particular, it will interact with the Instrument, Control, Processing, and Knowledge

Networks, and the CEI, to inform the direction of observations and to execute simulation tasks on the Grid. Furthermore, the Modeling Network will integrate into the user-configurable virtual observatory, and provide interface control with the observatory data systems and sensor simulator.