Extending Net-Centricity to Coalition Operations

Niranjan Suri, Andrzej Uszok, Rita Lenzi,
Massimiliano Marcon, Maggie Breedy, Jeffrey M. Bradshaw

Yat Fu, James Hanna, Vaughn Combs, Asher Sinclair,
Rob Grant, Robert Hillman

ihmc
FLORIDA INSTITUTE FOR HUMAN & MACHINE COGNITION

AIR FORCE RESEARCH LABORATORY

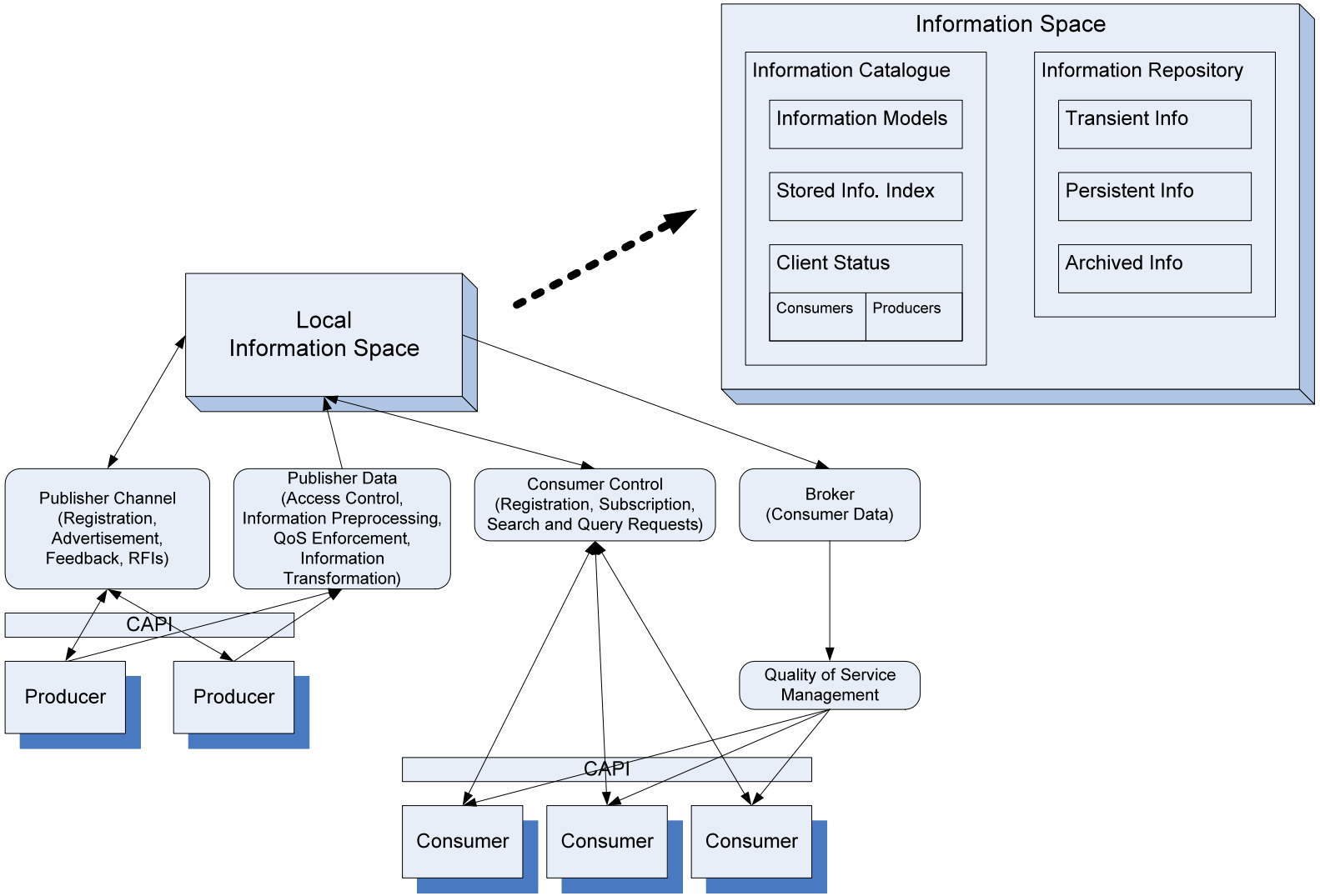
Motivation

- Network-centric Operations Considered Important
  - Information Superiority contributes to mission success
- Significant Research and Development within US DoD
- But… Most Operations are Coalition-based
- Therefore, need to support Network-centric Operations for Coalitions
- Challenge: Information Sharing in Coalition Environments

Example: Joint Battlespace Infosphere

- Architecture developed by AFRL
- Supports Publish / Subscribe / Query of Metadata tagged information
  - Handles information matchmaking, routing among multiple publishers and subscribers
  - Metadata expressed as XML
  - Subscriptions can use predicates
  - Supports queries over XML metadata
  - Supports archiving of published data
- Numerous implementations
  - AFRL: Apollo, Phoenix
    - Phoenix is also an abstract architecture – Fawkes is the first implementation
  - General Dynamics – Mercury

JBI Architecture



Requirements for JBI

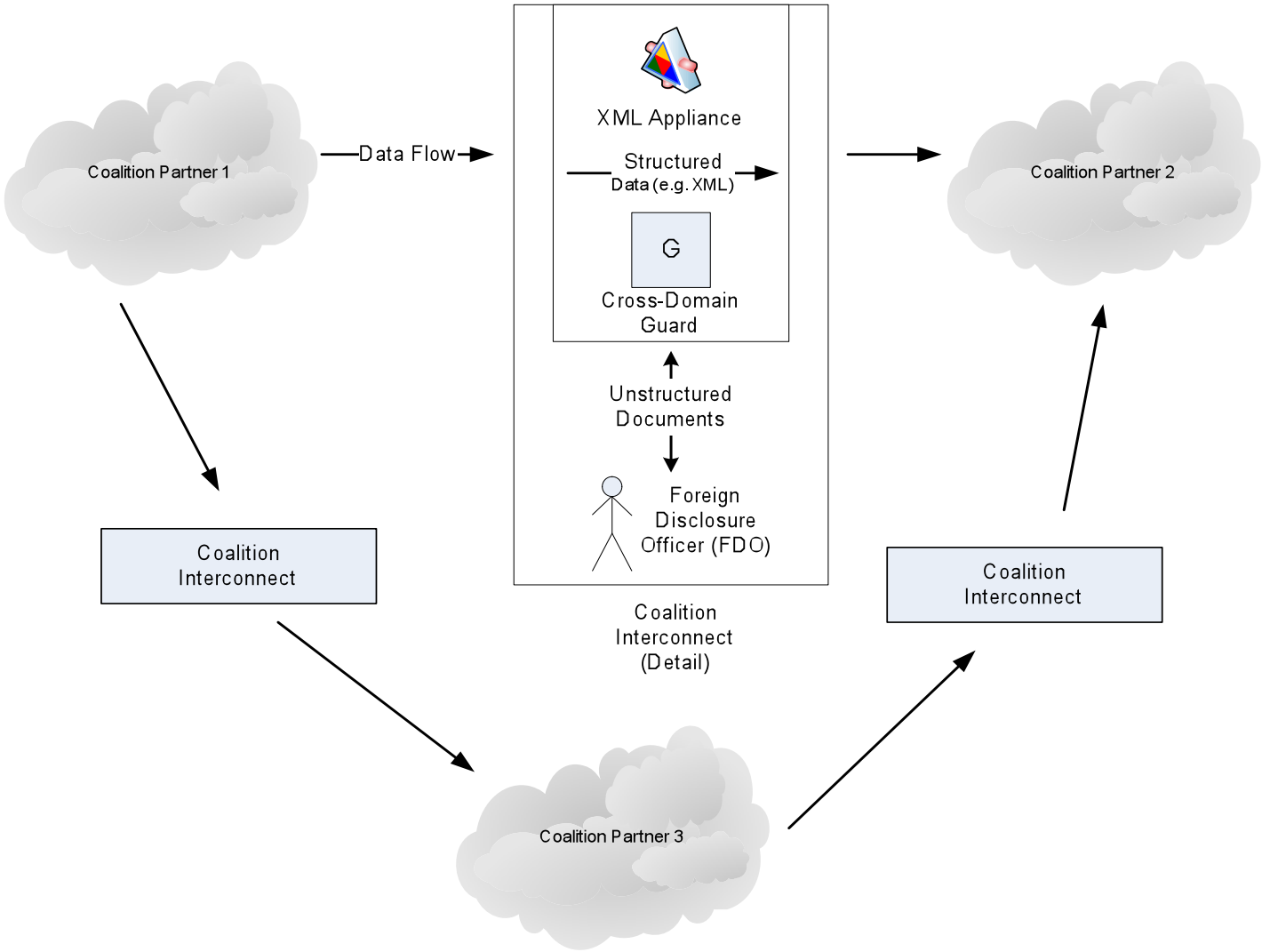
- Clients connect via the CAPI (Client API) to
  - Authenticate
  - Publish Information
  - Subscribe for Information
  - Query for Information

- In coalition settings, this would imply
  - Common authentication mechanisms
  - Network connectivity!
  - On-demand information exchange!!

Current State of Coalition Information Sharing

- Much Like Cross-Domain Information Sharing / Exchange (CDIS / CDIX)
- No Direct Network Connections Allowed
- All Data Must Flow Through Gateways / Interconnects
- Gateways use (Hardware) Guards
  - High-assurance, trusted, and hardened platform
    - For Example – Radiant Mercury
  - Preconfigured to Support Limited and Controlled Data Flows
  - Difficult / laborious to change

Current State of Coalition Information Sharing (Continued)



Problems with Current Solution

- Rigidity
  - Guards only allow pre-defined, structured data to pass
  - Changing policies in the Guard is difficult / time consuming
- Speed
  - Unstructured documents (or new types of structured documents) must undergo human review
- Opacity
  - Difficult / Impossible to Explore / Search for Information Across Coalition Boundaries

- Implies no Net-Centricity

Towards a Solution…

- AFRL's Services-based Phoenix IM Architecture
- AFRL's Cross-Domain Information Solution

- IHMC's Federation Capabilities
- IHMC's Policy Management Capabilities

Phoenix

Background

- The Apollo reference implementation is the culmination of several years of information management research
  - The Apollo architecture was not designed with SoA in mind
- The movement toward and availability of SoA based middleware permeates DoD
  - IM application of these technologies are little understood, utilized solely for information routing
- SoA's offer numerous advantages
  - Dynamic composition
  - Extensibility
  - Ability to rapidly address change requirements
- Needed a coherent and consistent architecture to support IM in a SOA

What is Phoenix?

- Service Oriented Architecture (SOA) for Information Management (IM)
  - Provides a set of independent, flexibly deployable IM services
    - Submission, Subscription, Information Brokering, Dissemination, Repository, Query, Type Management, Event Notification, Service Brokering, Session Management, Information Discovery, Security, Client Runtime, Connection, Stream Brokering, Stream Discovery, Stream Repository
  - Provides a set of supporting constructs
    - Information, Frame, Stream, Event, Channel, Filter, Session
  - Supports multiple orchestrations (reliability, availability, performance)
- Defines universal IM services (Pub/Sub/Query)

Constructs

- Information
  - Well characterized data that flows between and among producers and consumers (applications) and services
  - This construct consists of:
    - An information type identifier – Defines the well known structure of an information instance
    - Metadata – Describes the payload and is used for brokering (conforms to the metadata schema for this type)
    - Payload – The actual information (or reference)
    - An information context construct – Attributes that further describe the information instance and/or implementation specific actions

Constructs (cont.)

- Channel
  - Provides the mechanism for information to be moved between and among the producers, consumers, and services (entities)
    - Provides the "plumbing" that connects entities and enables effective and efficient information flow
    - Abstracts and encapsulates transport protocols
    - Segregates information and control flows
      - Control channel and Information channel is are distinct abstractions
      - May be implemented using different protocols

Constructs (cont.)

- Filters
  - Provide a mechanism to manipulate and/or modify information as it flows through channels
    - Filters may be attached to either end of a channel and may also be chained (composed)
    - Filters might be used to shape information flows to conform to Quality of Service (QOS) policy, to perform dirty word search/ scrubbing of information to conform to security policy, to multiplex/ de-multiplex information flows, etc.

Submission Service

- Accepts information provided through a Channel from a producing application
- Based on policy and service configuration
  - May pass the information to one or more Information Brokering Services through a Channel for predicate matching
  - May pass the information to one or more Repository Services through a Channel for information persistence

Information Brokering Service

- Matches the information against the set of registered predicates to determine all appropriate endpoint consumer applications
- Based on Configuration and policy:
  - May pass the information to one or more Dissemination services through a Channel for delivery to the appropriate endpoint consumer
  - May return a list of consumer IDs indicating appropriate endpoints for delivery

Dissemination Service

- Accepts information through a Channel
- Delivers the information to the appropriate endpoint consumer applications through a Channel
  - Based on the list of consumer IDs

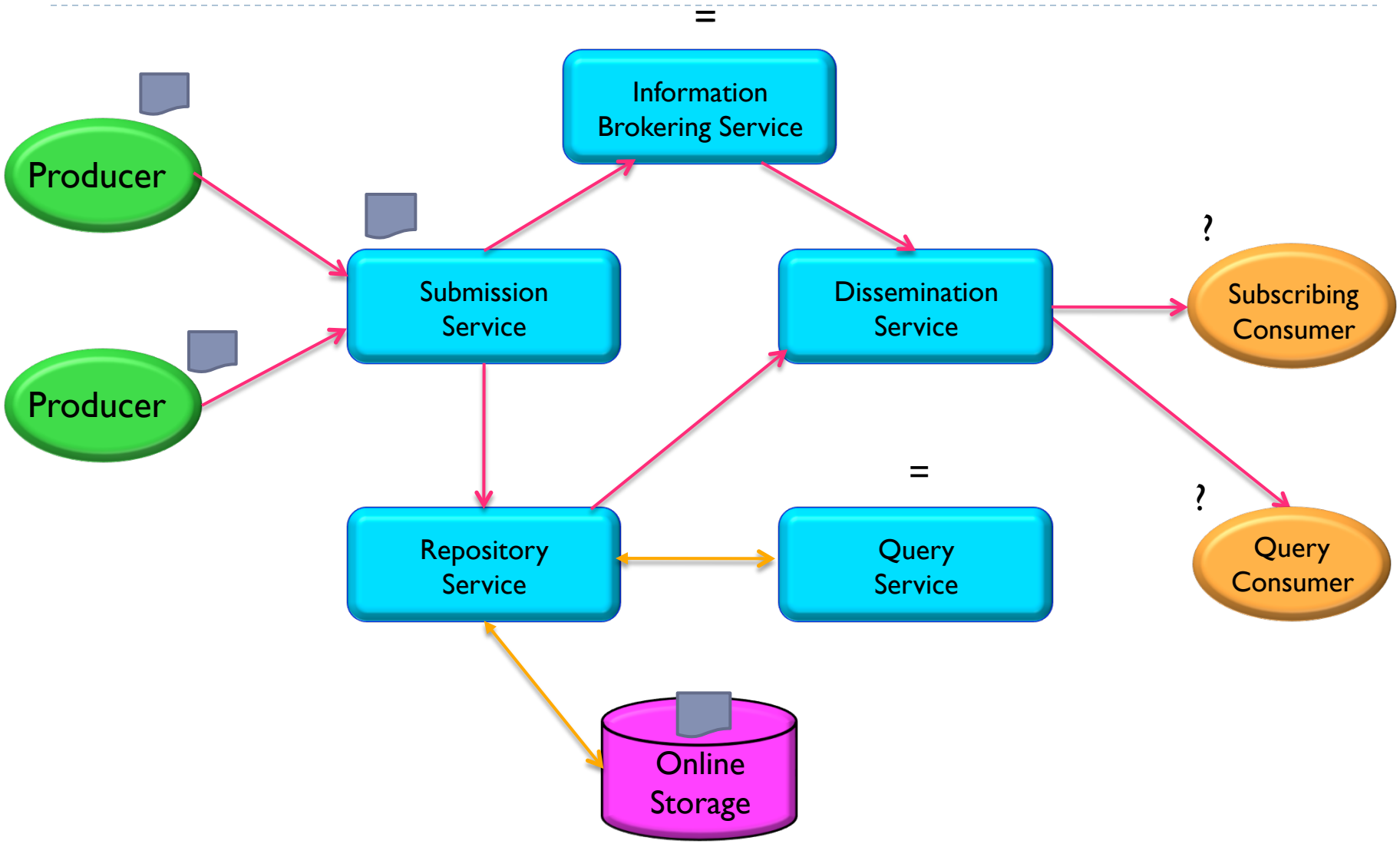
Repository Service

- Accepts Information through a Channel and inserts it into a data store
- Provides interfaces that enable the deletion of Information from the data store
- Provides interfaces that enable the archive and removal and of Information from the data store
  - Archives are higher latency data stores

Query Service

- Provides interfaces that enable information retrieval from one or more underlying data store(s)
- Delivers the Information to the appropriate consumer endpoint through a Channel
- Supports Synchronous and asynchronous query operations

Service Orchestration



Cross-domain Information Solution

Cross Domain Innovation & Science

- AFRL CDIS Group Building Solutions for CDS
- Approach Based on
  - XML Appliances
  - Cross Domain Guards
- Have Interconnected
  - Multiple Phoenix Instances
  - Static Information Flows Across Domains

Federation

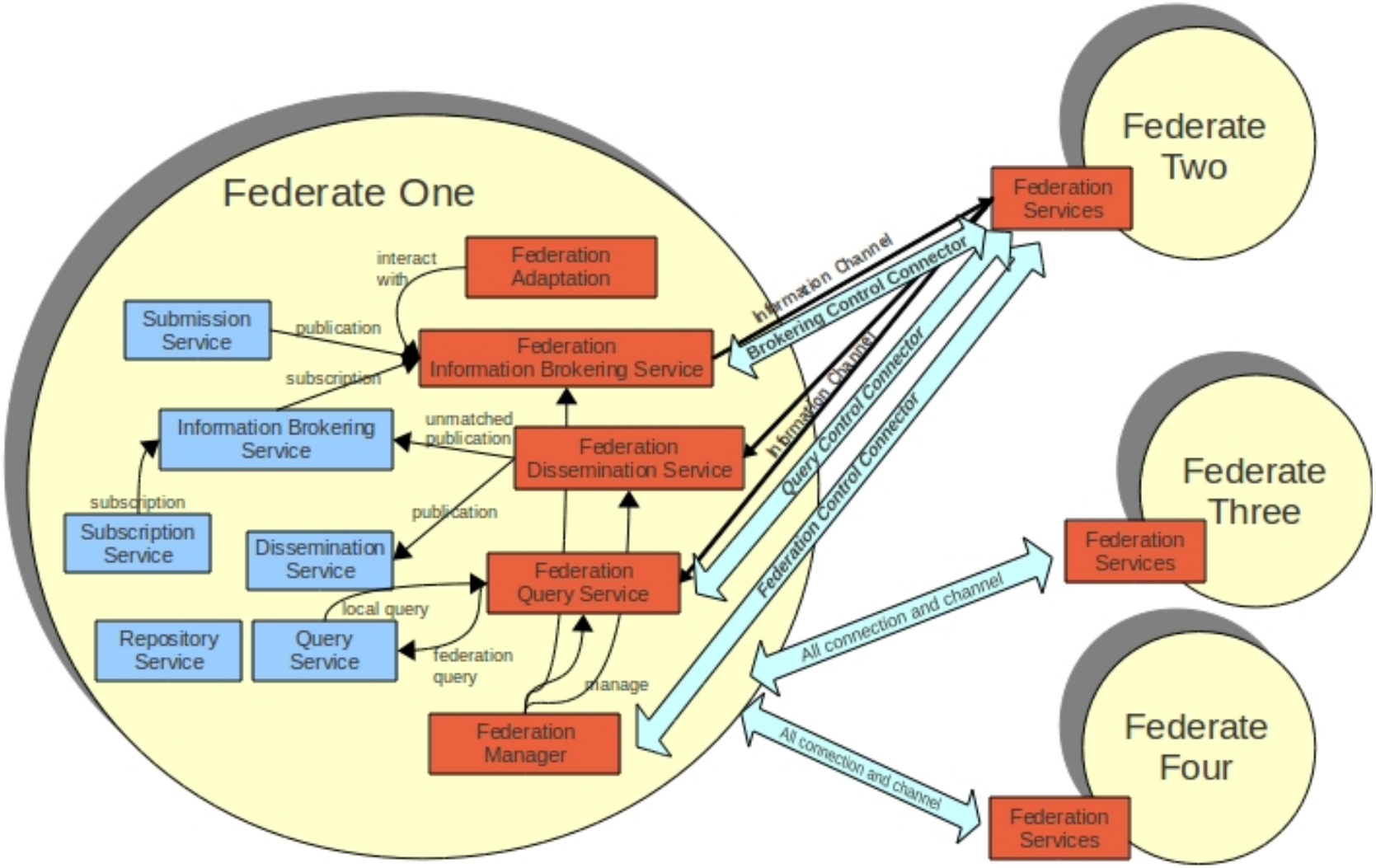
Federation Defined

- Assume there are multiple information enclaves
  - Collections of entities that can share information
  - Sharing defined as publish / subscribe / query
  - Sharing is not uncontrolled
    - Policies may regulate access to information
- JBI Perspective
  - Information enclave is called an InfoSpace
  - No overlap between InfoSpaces
    - That is, each client connects to one InfoSpace only
- Examples of InfoSpaces
  - Air Operations Center (AOC), Large UAV Platform, J-STARS, etc.

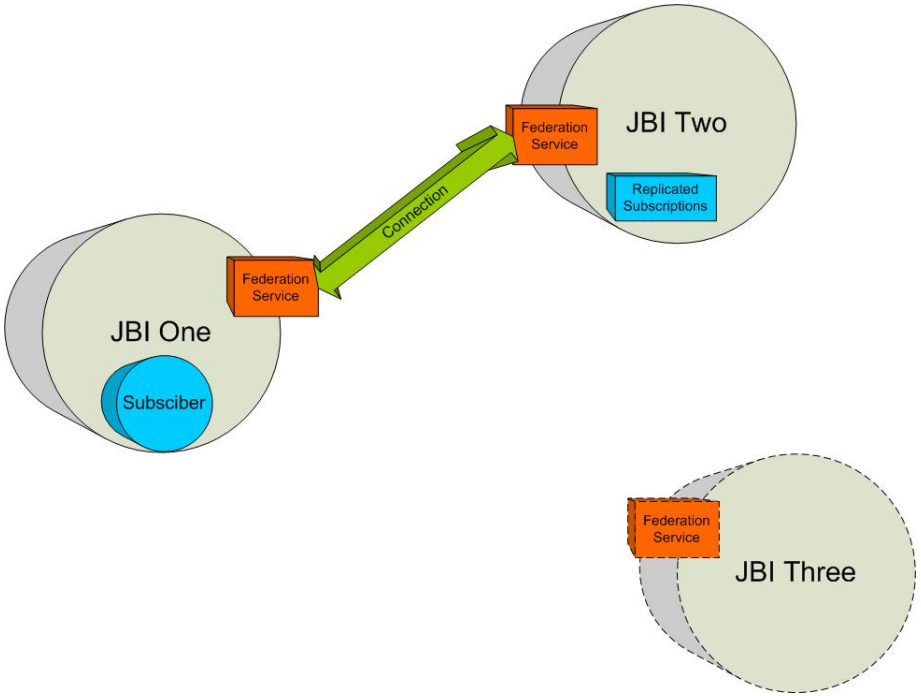
Federation Defined (Continued)

- Enable Interconnection Between Multiple InfoSpaces
- Interconnection is Peer-to-Peer
  - No master entity controlling federation
  - Federation is controlled independently from the perspective of each infospace
- Enable Sharing of Metadata / Information Across InfoSpaces
  - Seamless subscriptions and queries across infospaces
  - Transparency to clients
    - Client-Server connections / communication untouched
  - Controlled via policies – not unrestricted
  - Identity and integrity of individual infospaces preserved
- Efficiency when Handling Subscriptions and Queries
  - Criteria: Latency, Bandwidth, Storage, Availability, Resource Utilization
- Policy-based Control over Federation

Federation Architecture

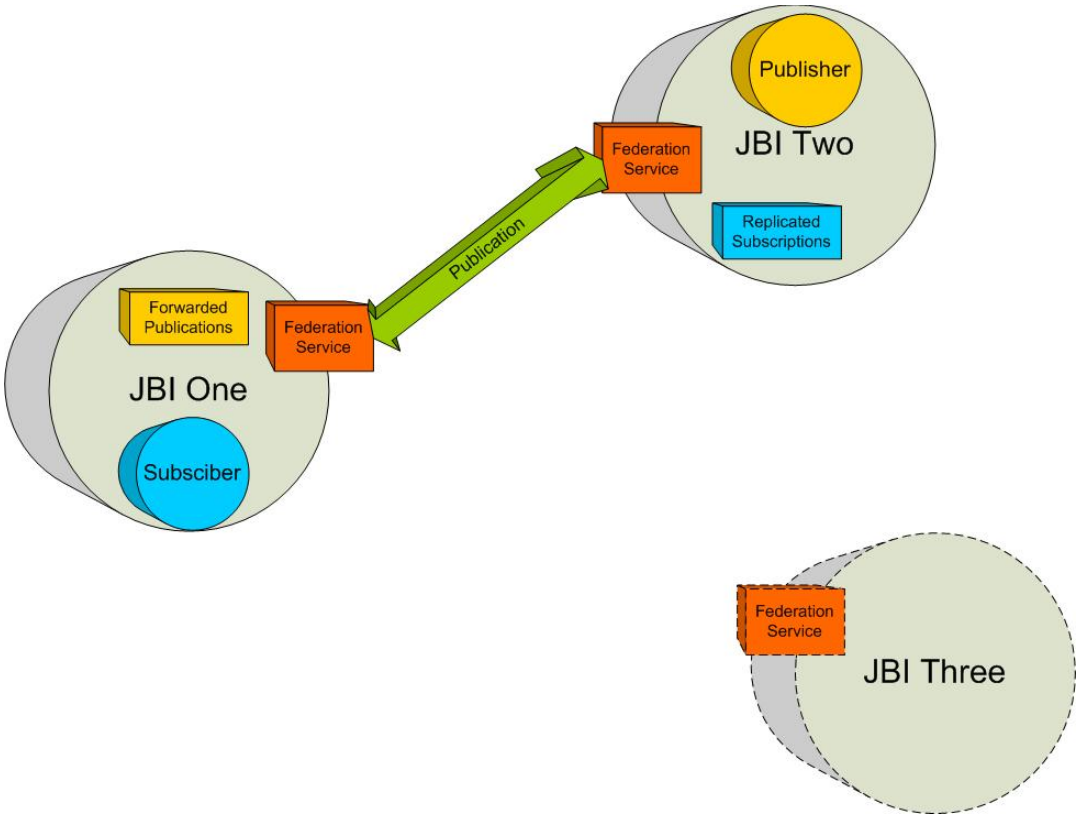


Establishing the Federation



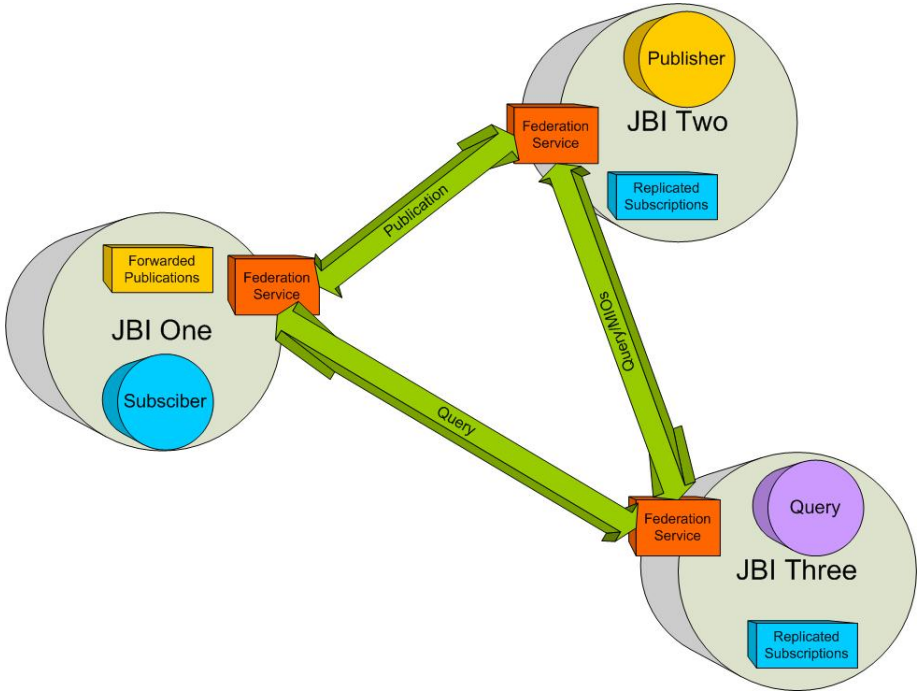
- Two JBIs discover each other and establish federation
- Subscriptions from the subscriber in *JBI One* are propagated through the federation and replicated in *JBI Two*

Publishing Across the Federation



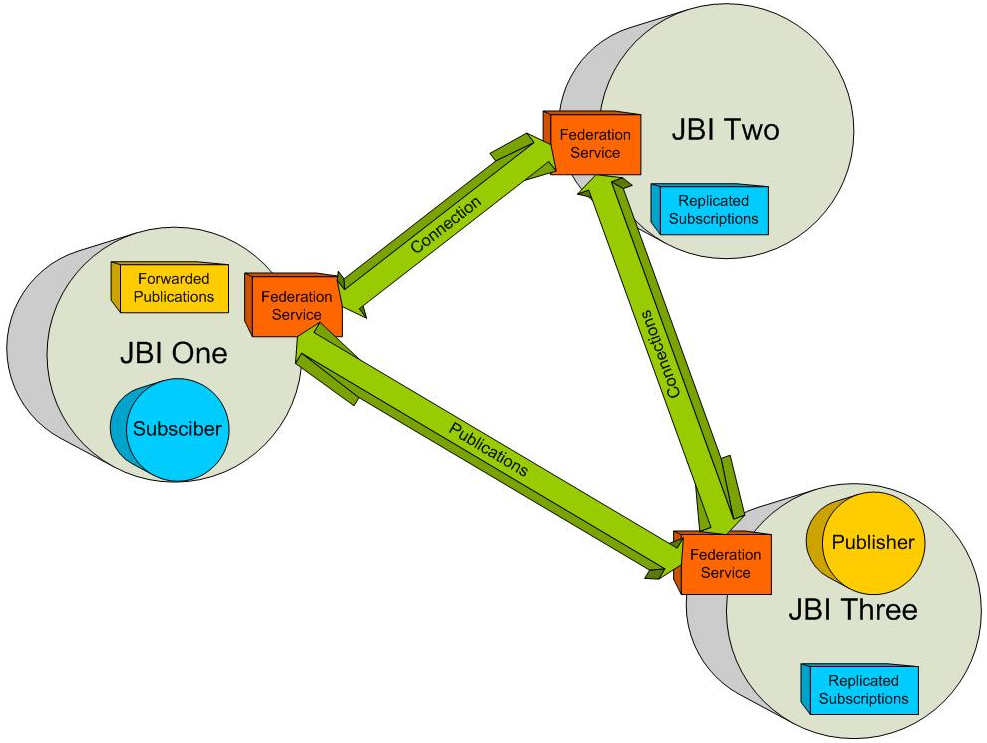
- When a publisher starts in *JBI Two*, any matching publications are propagated through the federation and delivered to the subscriber in *JBI One*

Expanding Federation and Query



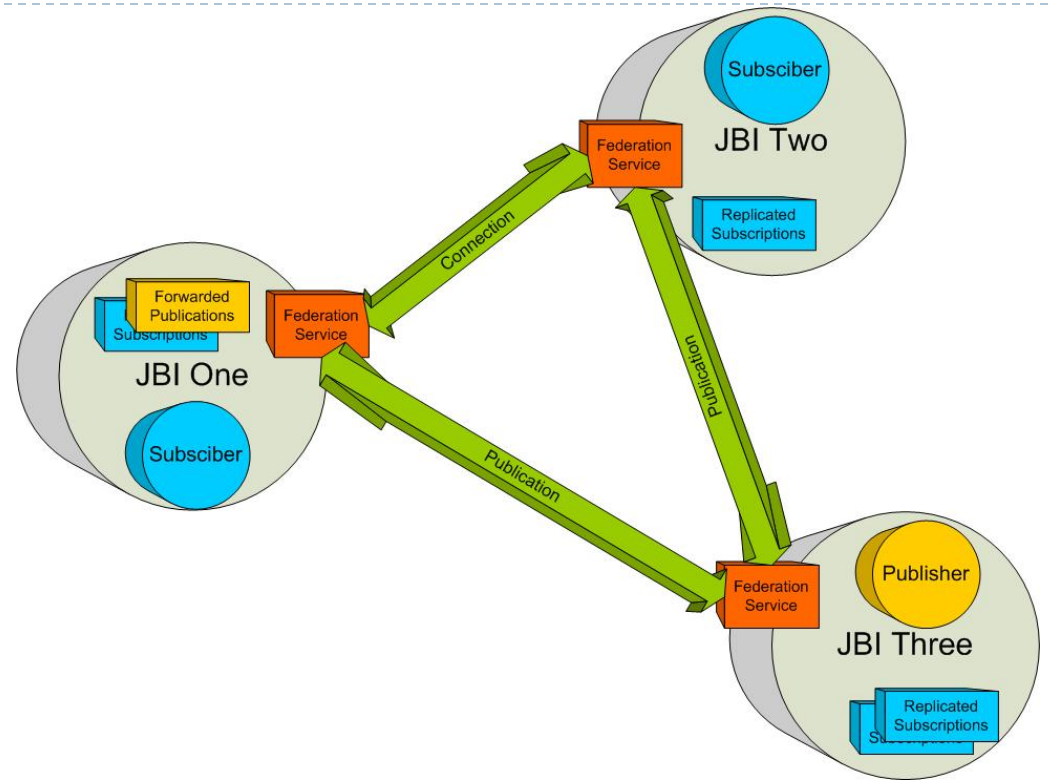
- A new federate (*JBI Three*) is started and discovered; it establishes connections with *JBI One* and *JBI Two*
- Existing subscriptions from *JBI One* are replicated in *JBI Three*
- Query client in *JBI Three* executes a query and receives MIOs from *JBI One*

Changes in Publishers



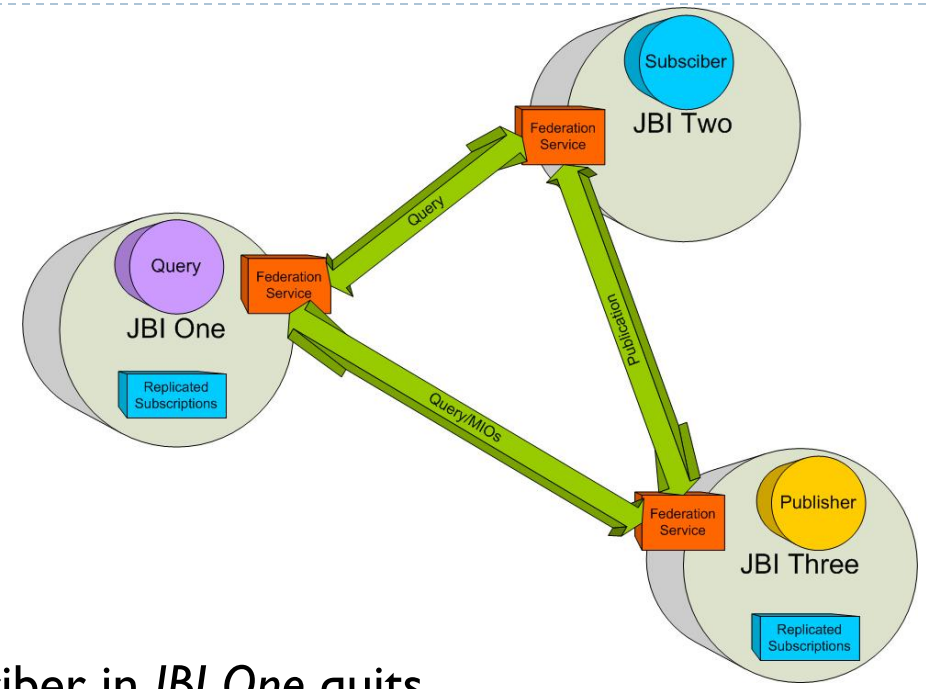
- The publisher in *JBI Two* quits, but, in the meantime, the new publisher in *JBI Three* has started publishing
- The subscriber in *JBI One* starts receiving publications from the new publisher

New Subscriber



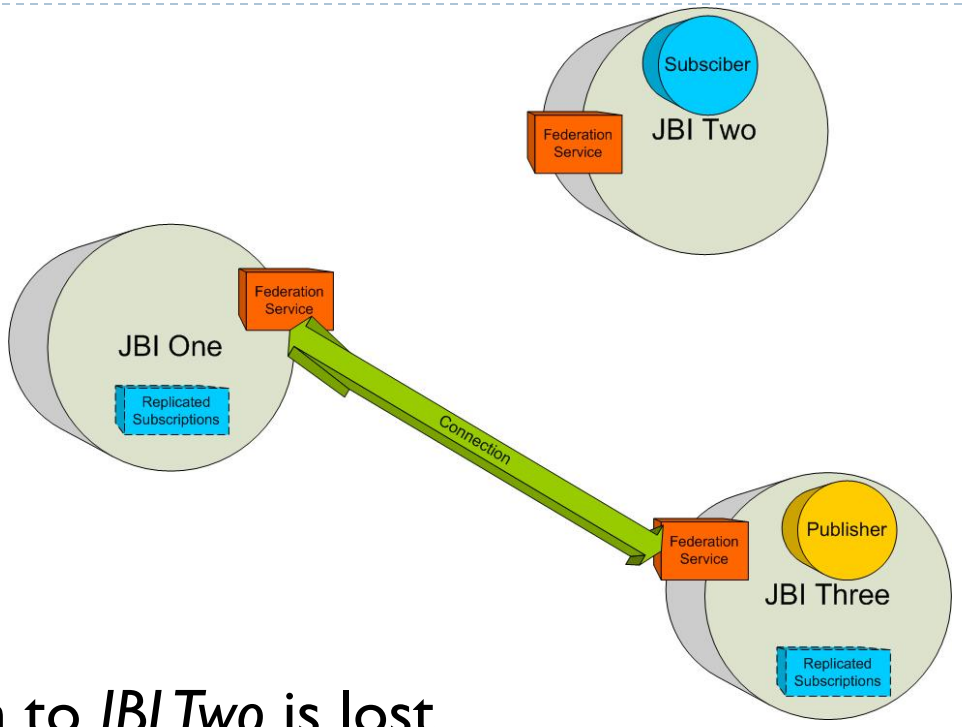
- A new subscriber registers in *JBI Two*
- Its subscriptions are replicated in *JBI One* and *JBI Three*
- The new subscriber starts to receive publications from the publisher in *JBI Three*

Subscriber Quits And New Query



- The subscriber in *JBI One* quits
- Its subscriptions are removed from *JBI Two* and *JBI Three*
- Since there are no longer any matching subscribers in *JBI One*, it stops receiving publications from *JBI Three*
- The subscriber in *JBI Two* keeps getting publications
- Query client in JBI One gets combined MIOs from JBI Three and Two

Disconnection of Federate



- Connection to *JBI Two* is lost
- After a delay if there is no reconnection from *JBI Two* its subscription are removed from *JBI One* and *JBI Three*
- Publication to *JBI Two* will stop immediately; in the future we plan to have store and forward protocol

Policy Management

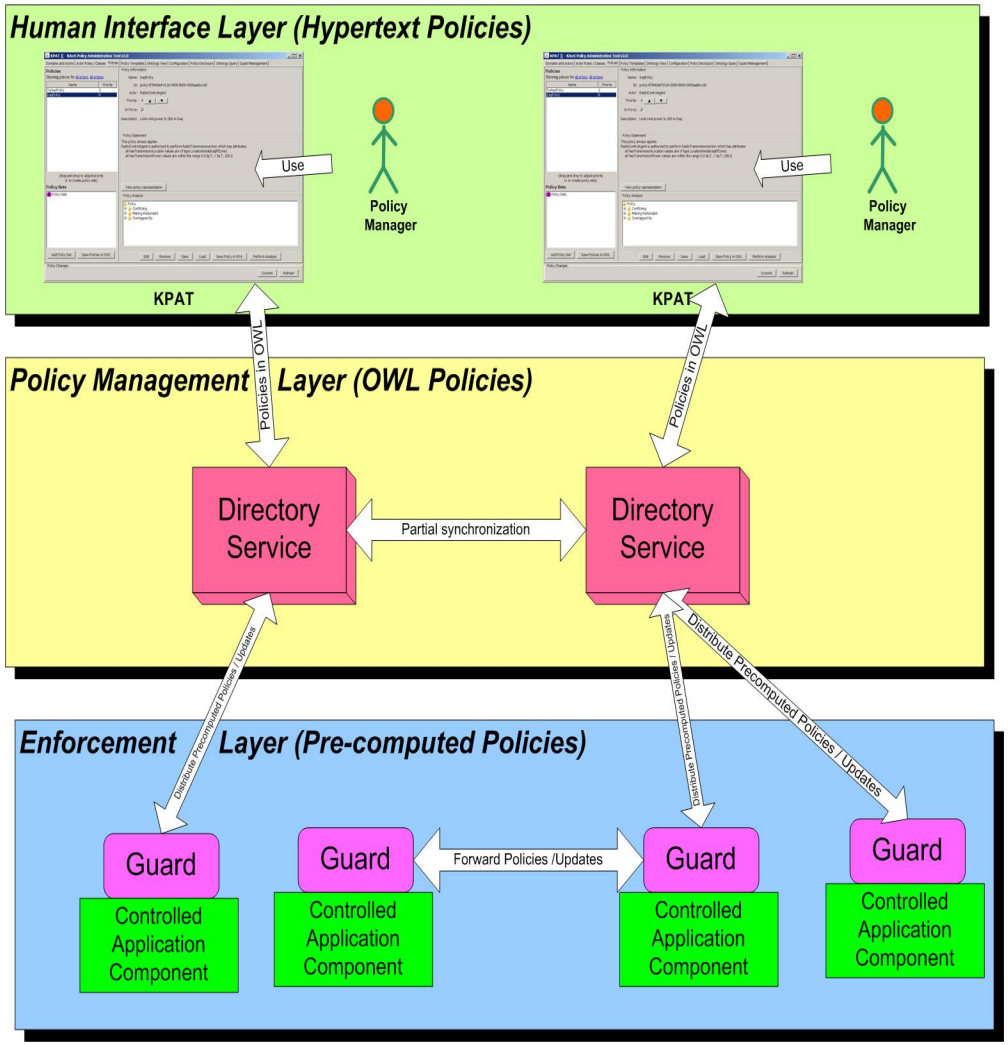
KAoS at a Glance

- Framework for policy and domain services
  - Allows policy-based governance of any aspect of system behavior. Enforces policy even for buggy, malicious, or non-compliant components
- Easily adapted for any agent, robot, or distributed computing platform through a Common Services Interface (CSI)
- Uses ontologies for policy, application components, and the real world
  - Uses W3C standard OWL, no "proprietary" language
  - Optional extensions to OWL expressiveness
  - Powerful and extremely efficient reasoning
    - Deontic logic by means of description logic
    - Incremental (non-monotonic) reasoning through snapshots, untell
    - "Compiled" to efficient runtime format so distributed guards continue enforcement even under disconnected operation
- KPAT: rich ontology-driven GUI for administration
- Kaa: KAoS adjustable autonomy and policy learning
  - Probabilistic reasoning about trust issues (e.g., GIG risk-adaptive access control)
  - Runtime adaptation of policies based on context-sensitive learning

For more information, see http://ontology.ihmc.us

Conceptual Architecture

- *Human interface (KPAT):* a point-and-click graphical interface for policy specification in the form of **natural English sentences**. The vocabulary is automatically provided from **ontology**.
- *Policy Management representation:* is used to encode and manage policy-related information in **OWL**. Inside DS it is used for policy analysis and deconfliction.
- *Policy Decision and Enforcement representation:* KAoS automatically "compiles" OWL policies to an **efficient lookup format** that provides the grounding of abstract ontology terms, connecting them to the instances in the runtime environment and to other policy-related information. Polices are sent from DS to **Guards**, which serve as local **policy decision points.**

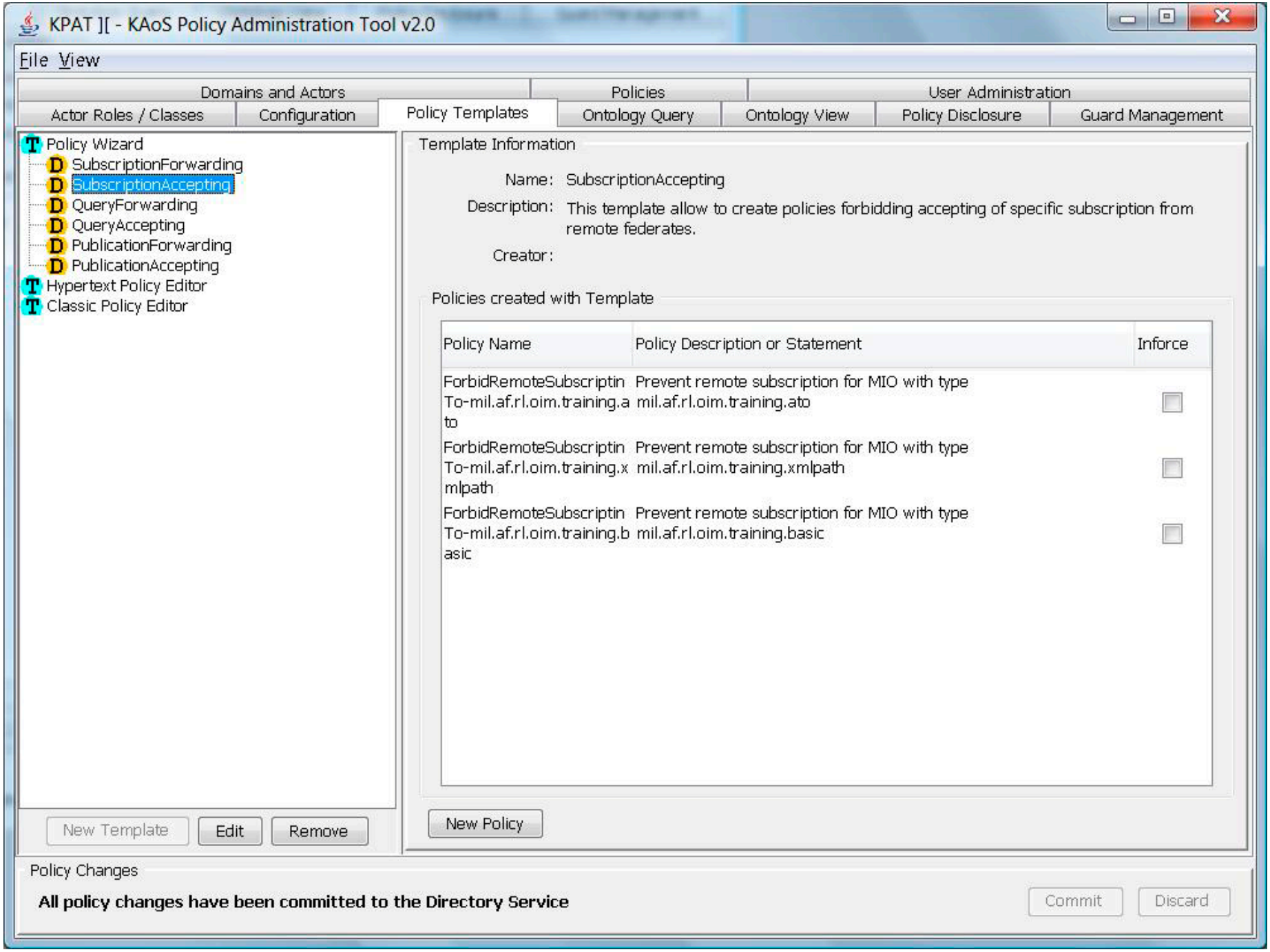


OWL Policy Syntax Example

Policy Example:

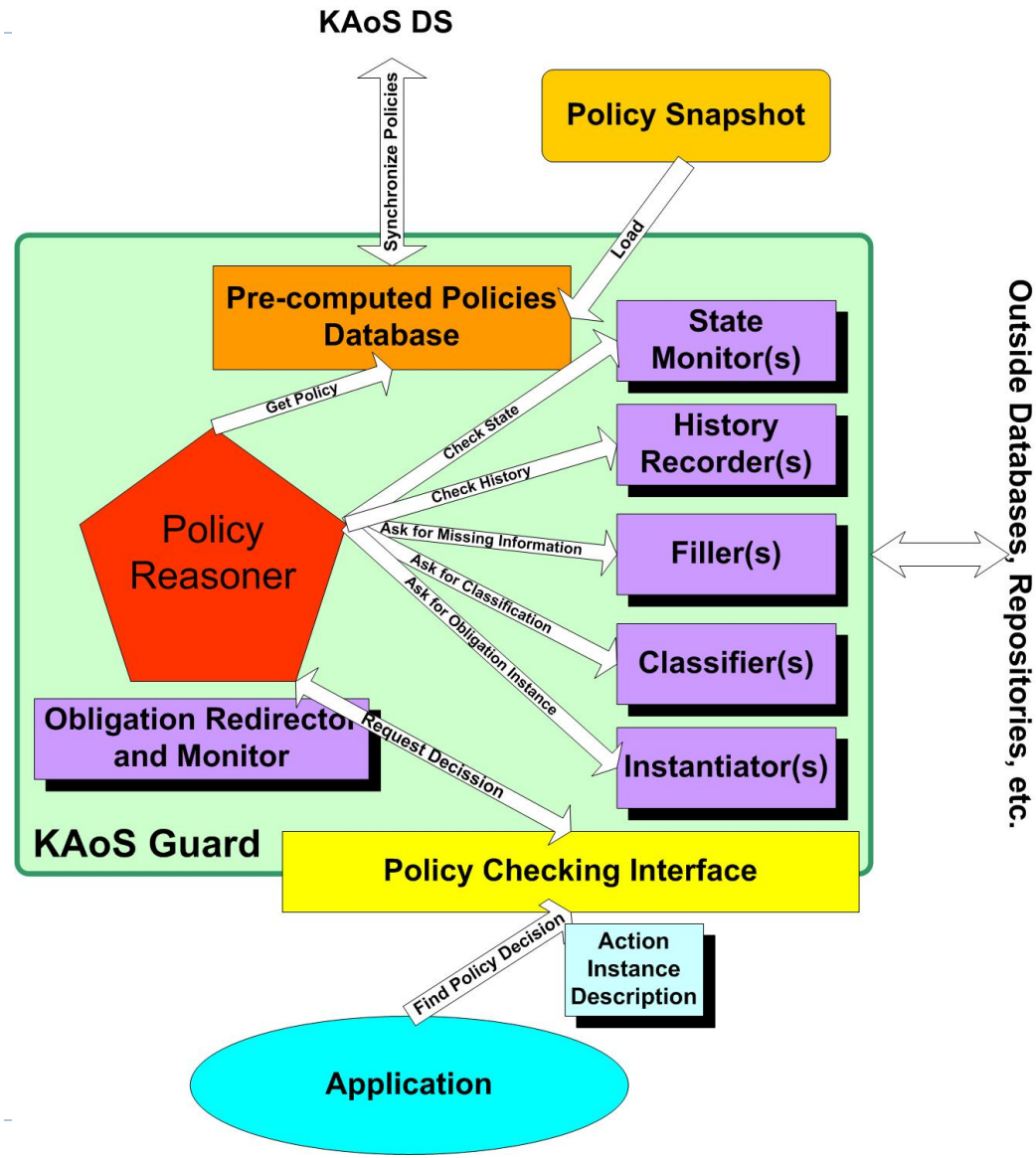
Any communication outside the Arabello domain, which is not encrypted is forbidden.

```
<?xml version="1.0" ?>
<!DOCTYPE P1 [
      <!ENTITY policy  "http://ontology.ihmc.us/Policy.owl#" >
      <!ENTITY action  "http://ontology.ihmc.us/Action.owl#" >
      <!ENTITY domains  "http://ontology.ihmc.us/ExamplePolicy/Domains.owl#" >
 ]>

<rdf:RDF
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
      xmlns:owl="http://www.owl.org/2001/03/owl+oil#"
      xmlns:policy="http://ontology.ihmc.us/Policy.owl#"
>
<owl:Ontology rdf:about="">
      <owl:versionInfo>$ http://ontology.ihmc.us/ExamplePolicy/ACP1.owl $</owl:versionInfo>
</owl:Ontology>

<owl:Class rdf:ID="OutsiteArabelloCommunicationAction">
      <owl:intersectionOf rdf:parseType="owl:collection">
            <owl:Class rdf:about="&action;NonEncryptedCommunicationAction" />
            <owl:Restriction>
                  <owl:onProperty rdf:resource="&action;#performedBy" />
                  <owl:toClass rdf:resource="&domains;MembersOfDomainArabello-HQ" />
            </owl:Restriction>
            <owl:Restriction>
                  <owl:onProperty rdf:resource="&action;#hasDestination" />
                  <owl:toClass rdf:resource="&domains;notMembersOfDomainArabello-HQ" />
            </owl:Restriction>
      </owl:intersectionOf>
</owl:Class>

<policy:NegAuthorizationPolicy rdf:ID="ArabelloCommunicationPolicy1">
      <policy:controls rdf:resource="#OutsiteArabelloCommunicationAction " />
      <policy:hasEnforcementSite rdf:resource="&policy;ActorSite" />
      <policy:hasPriority>10</policy:hasPriority>
      <policy:hasUpdateTimeStamp>446744445544</policy:hasUpdateTimeStamp>
</policy:NegAuthorizationPolicy>
```


KPAT][- KAoS Policy Administration Tool v2.0
File View
Domains and Actors
Policies
User Administration
Actor Roles / Classes
Configuration
Policy Templates
Ontology Query
Ontology View
Policy Disclosure
Guard Management
Policy Wizard
SubscriptionForwarding
SubscriptionAccepting
QueryForwarding
QueryAccepting
PublicationForwarding
PublicationAccepting
Hypertext Policy Editor
Classic Policy Editor
New Template
Edit
Remove
Template Information
Name: SubscriptionAccepting
Description: This template allow to create policies forbidding accepting of specific subscription from remote federates.
Creator:
Policies created with Template
Policy Name
Policy Description or Statement
Inforce
ForbidRemoteSubscriptinTo-mil.af.rl.oim.training.ato
Prevent remote subscription for MIO with type mil.af.rl.oim.training.ato
ForbidRemoteSubscriptinTo-mil.af.rl.oim.training.xmlpath
Prevent remote subscription for MIO with type mil.af.rl.oim.training.xmlpath
ForbidRemoteSubscriptinTo-mil.af.rl.oim.training.basic
Prevent remote subscription for MIO with type mil.af.rl.oim.training.basic
New Policy
Policy Changes
All policy changes have been committed to the Directory Service
Commit
Discard

KAoS Guard

- Where KAoS meets the application - policy decision point
- Policy checking traverses the policy database in policy priority order and checks to see whether the AID is in the range of actions controlled by any policy
  - the range of actions attribute is derived from an action class controlled by the policy,
  - role-value map relations, defining aspects of policy context, are checked as well.



Federation Policies

- Federation Acceptance Policies
  - E.g., whether to federate, and what priority and resource privileges should be given the federate
- Gatekeeping Policies
  - E.g., access control for a given federate
- Adaptation Policies
  - How the federation will adapt if resource requests outstrip availability
- Contract Policies
  - Govern the automated contract negotiation proecess

Conclusions / Future Work

Conclusions / Future Work

- Extending Information Management Capabilities to Coalitions Would be Valuable
- Multiple roadblocks
  - Some policy, some technical
- Technical Solutions Exist that can be Leveraged – Cross Domain Guard
- But – still need to operate within the restrictive environment
- Thoughts / Ideas
  - Easier to accredit SoA-based approaches (after changes)
  - How much flexibility can we have? (Or can we get away with?)