Supporting Federated Information Spaces with the Joint Battlespace Infosphere (JBI) Platform

[1]Niranjan Suri, [1]Andrzej Uszok, [1]Jeffrey Bradshaw, [1]Maggie Breedy, [1]Marco Carvalho,
[2]James Hanna, [2]Robert Hillman, [1]Tim Hutcheson, [1]Massimiliano Marcon, [2]Asher Sinclair

[1]Florida Institute for Human & Machine Cognition
Pensacola, Florida, USA

[2]Air Force Research Laboratory
Rome, New York, USA

Abstract

Proprietary and stove-piped information systems have slowly given way to standardized information management architectures such as the Joint Battlespace Infosphere (JBI) architecture developed by the US Air Force Research Laboratory. However, each independent coalition organization, branch, and/or mission is normally associated with a separate instance of a managed information space that operates in an independent manner. While this is necessary given the different stakeholders and administrative domains, the demands for coordination and cooperation require interoperability and information exchange between these independently operating information spaces.

This paper describes a federated approach to interconnecting multiple information spaces to enable data interchange. We propose a set of interfaces to facilitate dynamic, runtime discovery and federation of information spaces. We also integrate with the KAoS policy and domain services framework to realize policy-based control over the federation and exchange of information. The key components that have been implemented include a Federation Service that manages the life-cycle of discovering, negotiating, and managing a federation, as well as a Federation Connector, that manages the interconnection, translation, and wire protocols. Our approach allows clients to transparently perform publish, subscribe, and query operations across all the federated information spaces. We have integrated with two existing JBI-compliant implementations – Apollo from the Air Force Research Laboratory and Mercury from General Dynamics.

1. Introduction

Information systems are a key component of any military mission and are essential to ensuring their successful execution. Traditionally, information management was supported by stove-piped systems that were difficult to update, modify, and integrate. As a reaction to this problem, the US Air Force Research Laboratory developed the Joint Battlespace Infosphere (JBI) architecture [1] [2] [3]. JBI is a standardized information management architecture with multiple implementations and provides a flexible publish/subscribe/query model for information management.

The JBI architecture standardizes the interfaces for client applications (CAPI) to facilitate client integration into a JBI implementation. However, military missions often involve multiple nations and multiple forces, with each independent organization normally associated with a separate instance of a managed information space that operates in an independent manner. Such an organization is often required given the different stakeholders and administrative domains. However, the demands for coordination and cooperation require information exchange between these independently operating information spaces.

Federation supports the interconnection of multiple, independently managed information spaces in order to share information. Federation is a key enabling technology for multiple forces of a single nation as well as coalition forces to be able to cooperate together in support of a mission. This paper describes our approach to federation. We propose a set of interfaces to facilitate dynamic, runtime discovery and federation of infospaces. We also integrate with the KAoS policy and domain services framework to realize policy-based control over the federation and exchange of information. Our approach allows clients to transparently perform publish, subscribe, and query operations across all the federated information spaces.

The rest of this paper is organized as follows. Section 2 presents a short overview of JBI in order to facilitate the discussion of federation. Section 3 presents our overall architecture for federation. Section 4 describes the interfaces for federation. Section 5 presents the notion of federation service contracts and policy-based control. Section 6 presents some initial experimental results. Finally, Section 7 discusses related work and future work.

2. Overview of JBI

The architecture and motivations for JBI are described in detail in [2], which presents a reference model for Information Management. For the purposes of this paper, the essential elements of the JBI architecture are highlighted in Figure 1 below.

An Information Space is may be defined as one instance of a JBI-compliant system, which facilitates exchange of information between clients. A number of clients connect to the system and can behave as producers of information, consumers of information, or both.

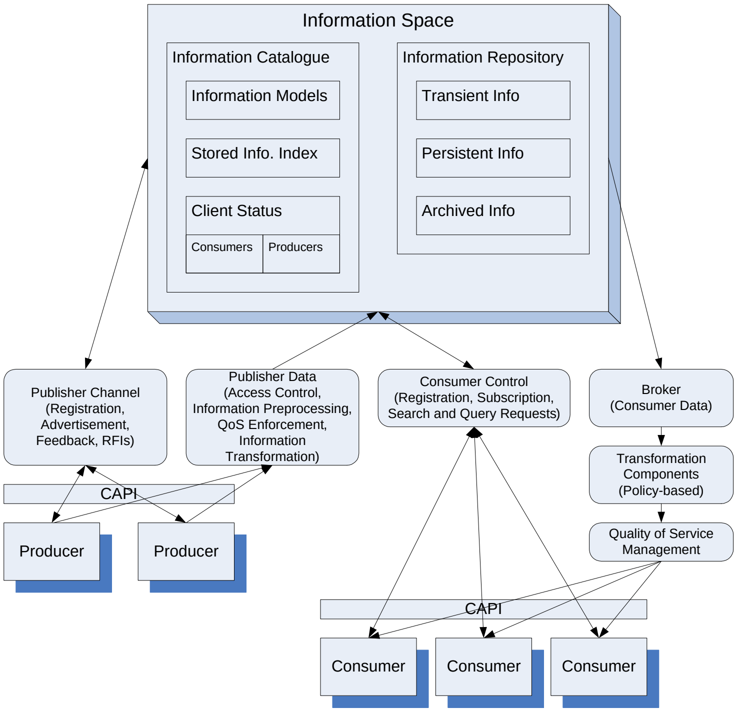

Information Space
Information Catalogue
Information Models
Stored Info. Index
Client Status
Consumers
Producers
Information Repository
Transient Info
Persistent Info
Archived Info
Publisher Channel (Registration, Advertisement, Feedback, RFIs)
Publisher Data (Access Control, Information Preprocessing, QoS Enforcement, Information Transformation)
Consumer Control (Registration, Subscription, Search and Query Requests)
Broker (Consumer Data)
Transformation Components (Policy-based)
Quality of Service Management
CAPI
Producer
Producer
CAPI
Consumer
Consumer
Consumer

Figure 1: Architecture of a JBI-Oriented Information Management System

The system includes both an Information Catalogue, which is a directory of information types known to the system, as well as an Information Repository, which handles the actual data. The Information Repository may optionally archive information for later retrieval using queries. Different implementations of a JBI-compliant system are free to use any approach as long as they comply with the syntax and semantics of the CAPI - the Client API. In the case of Apollo, one of AFRL's reference implementations of the JBI, the Information Catalogue is called the Metadata Repository (MDR), while the Information

Repository is called the Information Object Repository (IOR). Published data is represented as a Managed Information Object (MIO). Each MIO has a corresponding data type that is registered in the MDR, metadata in the form of an XML document, and a payload. Clients may have standing subscriptions based on the type, with an optional predicate to match against published metadata. If a predicate is specified, it is in the form of an XPATH expression, which can filter out unnecessary MIOs that a client is not interested in receiving. Clients may also execute queries, which results in matching MIOs being retrieved from the IOR and returned to the client.

A client typically connects to one (and only one) Information Space. While it is possible to connect to multiple information spaces, doing so places the onus on the client to discover the information spaces and connect to each one. The client would also need to be authenticated with multiple information spaces, which implies that all of them must have accounts for the client (difficult when there are multiple administrative domains involved). One of the benefits of Federation is to hide the presence of multiple information spaces from the clients. Each client continues to connect to one and only one information space, but has access to all allowed (controlled by policy) information across multiple information spaces.

3. Federation Architecture

The federation architecture supports seamless and secure integration of multiple information spaces, each of which is called a federate. Seamless implies that the architecture supports automatic discovery of and interconnection between federates. The process of federation is transparent to clients, which still connect to their home federate as normal. Secure implies that the federation process is not arbitrary and open. The establishment of federation and exchange of information is controlled via policies. Section 5 describes the role of policies in greater detail.

One important aspect of our federation architecture is that all the federates are peers. Each federate independently manages its connection with other federates. Each federate has its own set of policies that govern the exchange of information with other federates. This approach is logical given that each federate could potentially be in a separate administrative domain. However, we do envision that policies can be established from a single administration point if necessary, using the KPAT policy administration tool (see section 5). For example, consider a coalition operation involving three nations. Each nation controls its own information space via local policies, but the coalition headquarters may also impose policies that apply across all members of the coalition.

Figure 2 below shows our architecture for federation. The shaded boxes represent new component that have been added to the original architecture for an information space. The three major components are: a *Federation Service* (FS), a *Federation Connector* (FC), and *Dynamic Transformation Components* (DTC). Each federate has one instance of an FS. Each federate also has n instances of FCs, where n is the number of other federates that are part of the federation. That is, each FC instance handles the connection to one remote infospace. The DTCs are deployed as needed based on policy requirements.

The *Federation Service* (FS) handles the problem of *redirection*, as it manages data exchange between the infospace and its federates. As a prerequisite for interaction with federates, the manager of a given infosphere can configure its FS with information about other infospaces. This can include a set of policies that specify obligations and constraints on the behavior of the FS.

Note that the FS behaves both in the role of a consumer and producer with respect to other FS instances. In the role of a consumer, the FS will forward queries and predicates to receive remote MIOs whereas in the role of a producer, the FS will forward advertisements and locally produced MIOs to remote infospaces.

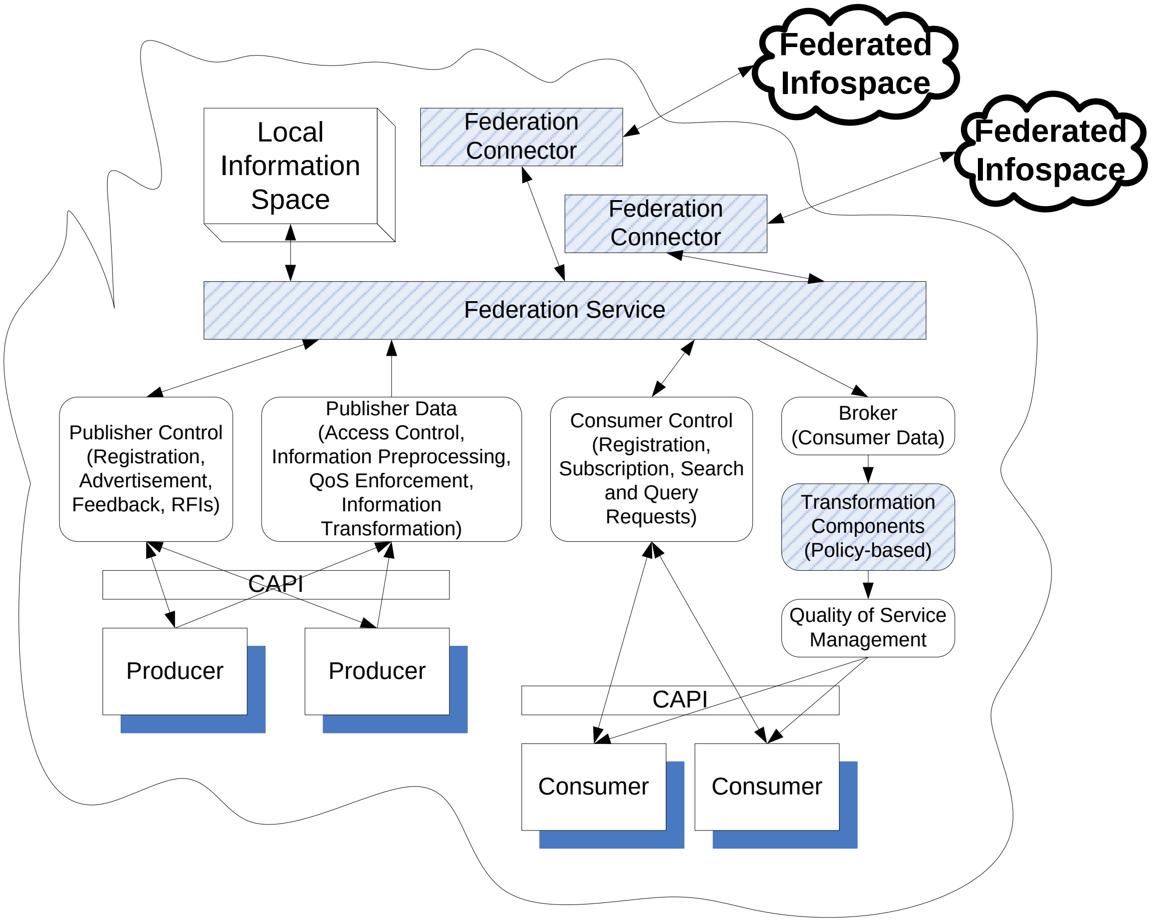

Federated Infospace
Federated Infospace
Local Information Space
Federation Connector
Federation Connector
Federation Service
Publisher Control (Registration, Advertisement, Feedback, RFIs)
Publisher Data (Access Control, Information Preprocessing, QoS Enforcement, Information Transformation)
Consumer Control (Registration, Subscription, Search and Query Requests)
Broker (Consumer Data)
Transformation Components (Policy-based)
Quality of Service Management
CAPI
Producer
Producer
CAPI
Consumer
Consumer

Figure 2: Architecture for Federation

4. Federation Interfaces and Implementation

An important aspect of this research effort has been the development of a generic set of interfaces that support federation. After examining the JBI architecture and two different implementations, Apollo from AFRL and Mercury from General Dynamics, we developed the following five key interfaces: `IMDService`, `InfoObjectReceptor`, `QueryReceptor`, `PredicateEvaluator`, and `AdaptationOracle`. These interfaces are implemented by various classes as described below. Figure 3 below shows the important components inside the *Federation Service*.

The `IMDService` supports all of the operations that one federate may want to execute on another federate. It is implemented by the *Federation Service* (FS) and is invoked by the local implementation of the information management system (IMS) – for example, Apollo. Each remote federate is represented by an instance of a *Remote Federation Service Proxy* (RFSP), which also implements the `IMDService` interface. Each proxy contains an instance of a *Federation Connector* (FC) and an instance of a *Remote Request Handler* (RRH), both of which also implement the same interface. The FC handles the network communication with the remote federate. The RRH receives incoming requests from the remote federate and executes them on the local IMS.

Consider the example of handling a new publication from a client. This results in the local information space invoking `newPublication()` on the FS. The FS invokes `newPublication()` on each of the active RFSP. If allowed by policy, and if the publication matches a remote subscription, the RFSP invokes `newPublication()` on the FC, which serializes and transmits the published object to the remote federate. The RFSP therefore acts as a Policy Enforcement Point (PEP). On the remote side, the FC receives the object and passes it to the RRH, using the `newPublication()` method again. The RRH on the remote side then injects the published object into the remote information space, where it is delivered to any relevant clients. Information objects are delivered via the `InfoObjectReceptor` interface, which is implemented by a modification made to the remote information space implementation. The process is inverted when a client of a remote federated publishes an information object that is received by the local federate.

In the case of a remote federate invoking a query, the query is executed by invoking the local IMS via the `QueryReceptor` interface. While not shown in the figure, the `PredicateEvaluator` interface is used when a predicate for a remote subscription needs to be evaluated. This is invoked by the RFSP, when a new publication is received, in the cases where a remote subscription has a predicate.

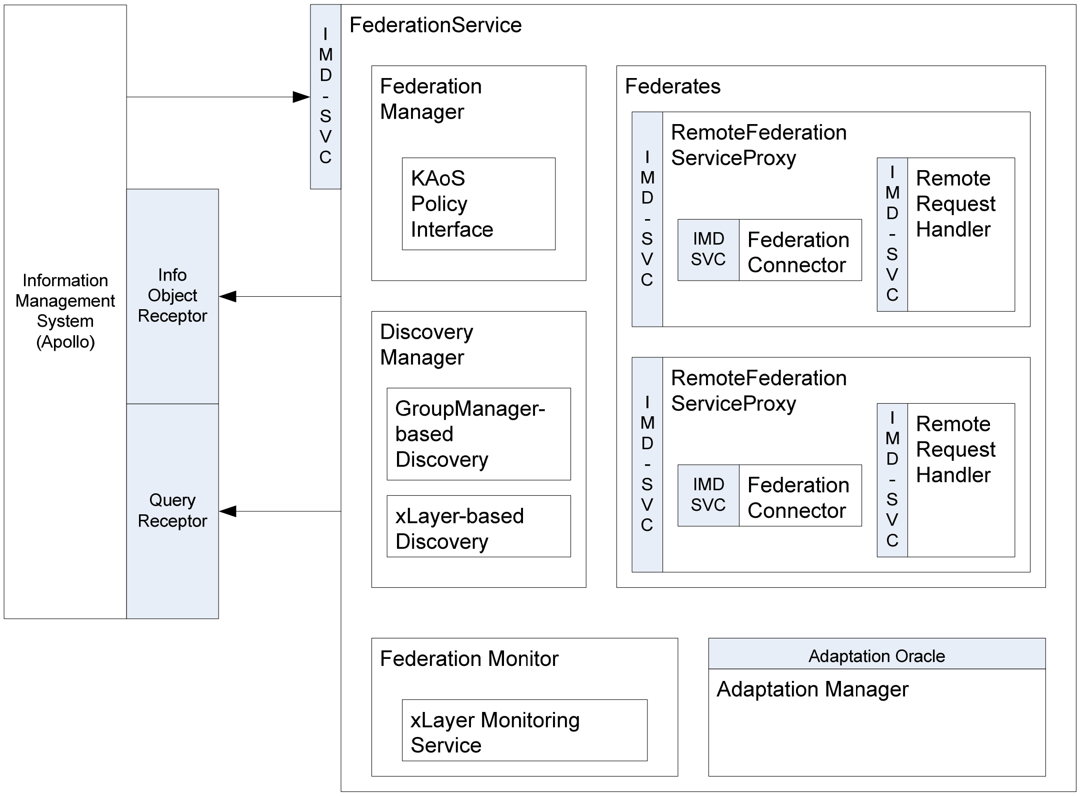


Figure 3: Federation Interfaces and Federation Service Architecture

The other major components shown in the figure are the *Discovery Manager* (CM) and the *Federation Manager* (FM). The DM handles the discovery of remote federates using two options – one based on the Group Manager [4] and the other based on the XLayer cross-layer substrate [5]. The XLayer substrate also provides a monitoring service that maintains detailed statistics and trends regarding the behavior of the network as well as the federation. For example, statistics such as CPU load, bandwidth utilized per connection to each remote federate, and the hit rate of remote predicates.

The final important component in the architecture is the *Adaptation Manager* (AM) and the `AdaptationOracle` interface. The AM automatically and dynamically changes the behavior of the federation to adapt to changing runtime conditions. The other components in the FS consult the AM via the `AdaptationOracle` interface.

Two specific adaptations have been implemented to date. The first adaptation is to handle a CPU-overload situation in the local federate. In such cases, predicate processing for remote subscriptions is suspended temporarily. The subscriptions are sorted based on the hit rate of their predicates and successively disabled until the CPU is no longer overloaded. Turning off local evaluation of remote predicates implies that all publications that match the type are sent to the remote federate. Predicates with a high hit rate (i.e., ones that match a large number of published objects) are selected first since disabling their evaluation increases the bandwidth utilized by the minimum amount possible.

The second adaption is to handle a network overload situation in the connection with a remote federate. In this case, the adaptation is to temporarily disable remote subscriptions entirely. The subscriptions are chosen based on their priorities, which are specified by the remote clients.

One final point worth noting is that the connections to the remote federates uses the Mockets communications library [6]. Mockets replace TCP sockets and provide the transport capabilities for the FCs. Mockets provide significant performance enhancements in wireless tactical environments by means of specific features such as bandwidth limitation, message replacement, prioritization, and detailed statistics.

The behavior of all the components in the Federation Service is dynamically controllable at runtime via policies. The next section describes the KAoS policy framework and its integration into federation.

5. Federation Service Contracts

One of the keys to coordinated operation of federated infospheres is a comprehensive, semantically-rich, and enforceable service agreement. The privileges and obligations of each infosphere within the federation must be established and monitored for compliance at all times. The service agreement binds all parties to act according to the constraints agreed to by the federation. This approach is absolutely necessary to ensure the proper flow of information through the federation. The KAoS Policy Service with specific extensions, integrated with InfoFed, is used to create and enforce federation contracts.

5.1 Technical Overview of the KAoS Services Framework

KAoS [7], a set of platform-independent services enables people to define policies ensuring adequate security, configuration, predictability, and controllability of both agents and traditional distributed systems. KAoS Domain Services provide the capability for groups of software components, people, resources, and other entities to be semantically described and structured into organizations of domains and subdomains to facilitate collaboration and external policy administration. KAoS Policy Services allow for the specification, management, conflict resolution, and enforcement of policies within domains. KAoS policies distinguish between *authorizations* (i.e., constraints that permit or forbid some action by an actor or group of actors in some context) and *obligations* (i.e., constraints that require some action to be performed when a state- or event-based trigger occurs, or else serve to waive such a requirement).

The use of ontology, encoded in OWL (Web Ontology Language, *http://www.w3.org/TR/owl-features/*), to represent policies enables reasoning about the controlled environment, about policy relations and disclosure, policy conflict resolution, as well as about domain structure and concepts. KAoS reasoning methods exploit description-logic-based subsumption and instance classification algorithms and, if necessary, controlled extensions to description logic (e.g., role-value maps).

KAoS Architecture. Two important requirements for the KAoS architecture have been modularity and extensibility. These requirements are supported through a framework with well-defined interfaces that can be extended, if necessary, with the components required to support application-specific policies. The basic elements of the KAoS architecture are shown in Figure 4; its three layers of functionality correspond to three different policy representations:

Human interface layer: This layer uses a hypertext-like graphical interface for policy specification in the form of natural English sentences. The vocabulary is automatically provided from the relevant ontologies, consisting of highly-reusable core concepts augmented by application-specific ones.

Policy Management layer: Within this layer, OWL is used to encode and manage policy-related information. The Distributed Directory Service (DDS) encapsulates a set of OWL reasoning mechanisms.

Policy Monitoring and Enforcement layer: KAoS automatically "compiles" OWL policies to an efficient format that can be used for monitoring and enforcement. This representation provides the grounding for abstract ontology terms, connecting them to the instances in the runtime environment and to other policy-related information (Figure 4).

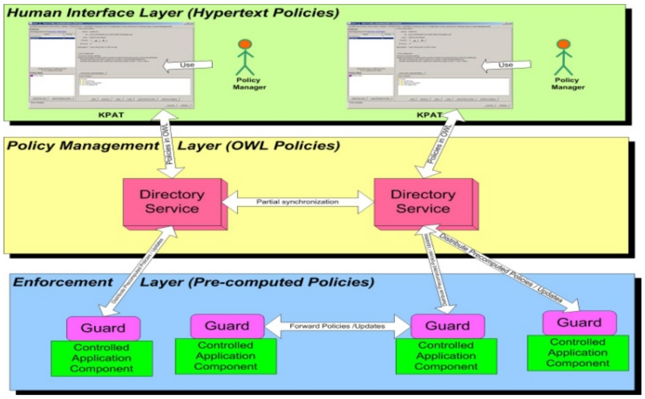
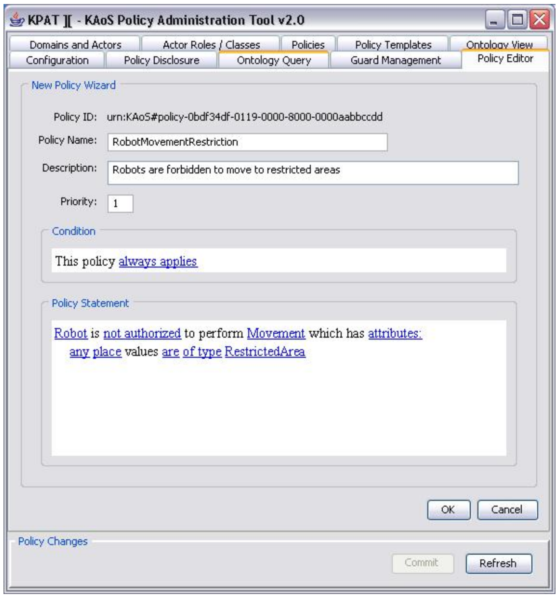

Human Interface Layer (Hypertext Policies)
Use
Policy Manager
KPAT
Use
Policy Manager
KPAT
Policy Management Layer (OWL Policies)
Policies in OWL
Policies in OWL
Directory Service
Partial synchronization
Directory Service
Distribute Precomputed Policies / Updates
Distribute Precomputed Policies / Updates
Distribute Precomputed Policies / Updates
Enforcement Layer (Pre-computed Policies)
Guard
Controlled Application Component
Guard
Controlled Application Component
Forward Policies /Updates
Guard
Controlled Application Component
Guard
Controlled Application Component

Figure 4: KAoS Policy Services Conceptual Architecture

KPAT, a graphical tool, allows specifying, analyzing, and modifying policies at runtime. KPAT hides the complexity of the OWL representation from users.



Policy ID: urn:KAoS#policy-0bdf34df-0119-0000-8000-0000aabbccdd

Policy Name: RobotMovementRestriction

Description: Robots are forbidden to move to restricted areas

Priority: 1

Condition

This policy always applies

Policy Statement

Robot is not authorized to perform Movement which has attributes:
any place values are of type RestrictedArea

Figure 5: Authorization Policy in the KPAT Hypertext Policy Editor

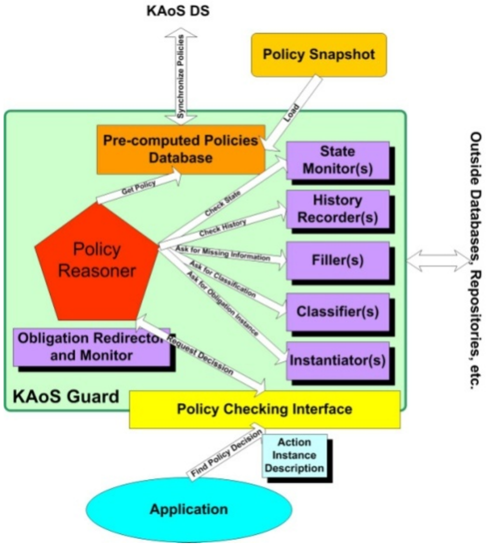


Figure 6: KAoS Guard – the policy decision point integrated with the application

5.2 Controlling Federation

The Federation Service in each of the federate is integrated with the KAoS Guard, which stores policies controlling establishment, lifecycle, information exchange and adaptation of the federations established by this federate. When the new potential federation partner is discovered and the initial connection it established then the given federate sends to its partner federate a set of information:

- List of its properties, such us ownership, mission, security clearance level, location, etc.
- List of metadata types the federate clients potentially intends to subscribe to or query about with importance priority attached to these metadata types,
- Matrix of values indicating preferences for using different possible adaptation methods on the connections between the federates.

Then each federate independently on its side decides, based on its own local policies:

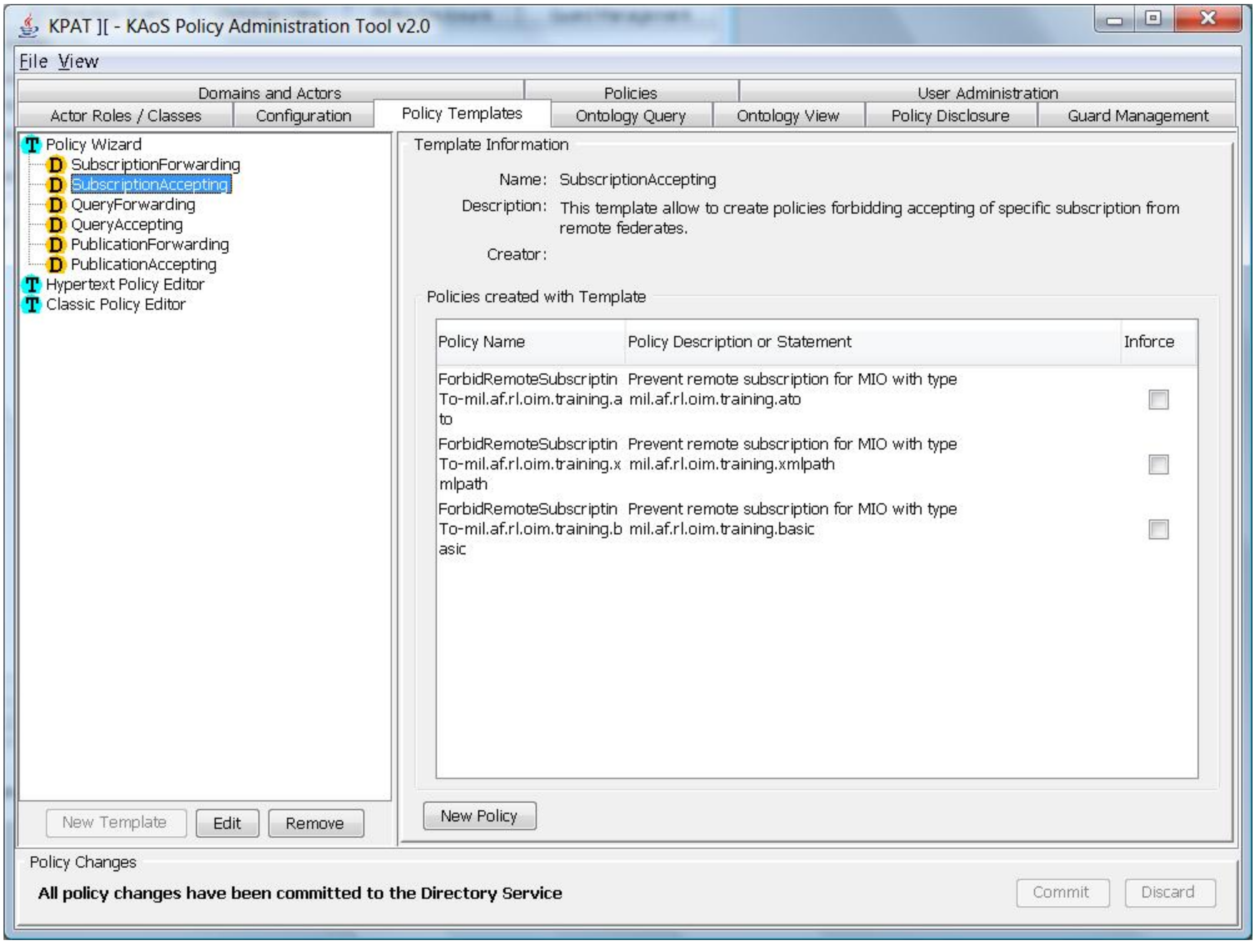
- Whether to establish federation with the remote federate,
- What priority to attach to the given remote federate,
- Estimate based on the current resource utilization for the federation operations and the assigned federate priority how much resources it can devote to server requests from the given federate (express as a percentage of time),
- Further it predicts what metadata type subscriptions or queries it would be able to realistically support for the given federate and sends this information to the remote federate.

During the subsequent exchanges of subscriptions, queries and publication between federates, each operation is examined and consulted with the current policies. They can forbid the given operations or modify it by changing the subscription or query predicate or trimming the metadata information in the published information object being forwarded to the remote federate.

Additionally policies and the agreed adaption matrix control when and which adaptation mechanism are activated when the share of resources used by the given federate exceeds the agreed limit.

The KAoS Guard is controlled (through the KAoS Directory Service) using KPAT, the graphical policy management tool. The KPAT configuration for the control of federation consists of sets of predefined policy template and polices associated with them (Figure 7). These policies can be easily activated and deactivated. The policy templates are groups into for menus:

- Federation Acceptance Polices,
- Gatekeeping Policies,
- Adaptation Policies,
- Contract Policies.



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	☐

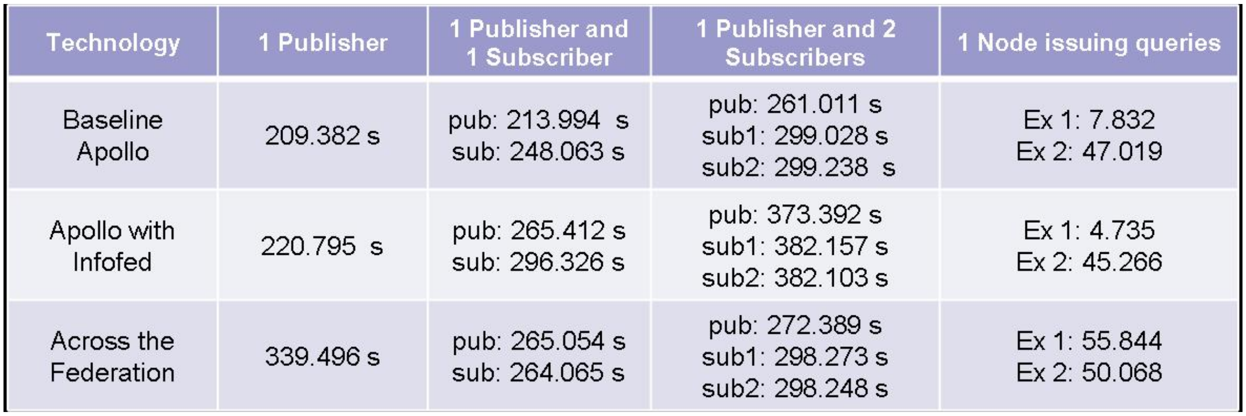
Policy Changes

All policy changes have been committed to the Directory Service

Figure 7: Federation Policy Templates in KPAT

6. Experimental Evaluation

The Federation Service has been evaluated to measure the overhead from inserting the Federation Service into the original Apollo-based JBI implementation and the performance of publish, subscribe, and query operations between locally attached clients versus clients attached to remote federates. The results obtained first prove the correctness and completeness of the Federation Service. Second the measured overhead and delays in operations performed across federation are adequate for the new functionality added to the JBI implementation.



Technology	1 Publisher	1 Publisher and 1 Subscriber	1 Publisher and 2 Subscribers	1 Node issuing queries
Baseline Apollo	209.382 s	pub: 213.994 s sub: 248.063 s	pub: 261.011 s sub1: 299.028 s sub2: 299.238 s	Ex 1: 7.832 Ex 2: 47.019
Apollo with Infofed	220.795 s	pub: 265.412 s sub: 296.326 s	pub: 373.392 s sub1: 382.157 s sub2: 382.103 s	Ex 1: 4.735 Ex 2: 45.266
Across the Federation	339.496 s	pub: 265.054 s sub: 264.065 s	pub: 272.389 s sub1: 298.273 s sub2: 298.248 s	Ex 1: 55.844 Ex 2: 50.068

Figure 8: Federation Service Performance Results

The table in Figure 8 shows results of different tests performed with the Federation Service. The Technology column shows the configuration of Apollo used during the test. The subsequent two columns show the time needed to perform publication and subscription of 20100 information objects, with one publisher and either one or two subscribers. The last column shows time needed to perform examples queries.

7. Related and Future Work

We are aware of only one other effort related to our work on federating information spaces – the ELSIF project [8]. ELSIF differs from our effort in a number of ways. In ELSIF, the developers treat the federation as something that is managed by a single entity (although the individual federates may be managed independently). In our work, federation is the result of a number of federates independently interconnecting in a peer-to-peer manner. Another major difference is that ELSIF works by wrapping the client API whereas our effort actually integrates the Federation Service into the JBI implementation, such as Apollo or Mercury.

Several interesting aspects of federation remain to be realized. For example, our current implementation works with Apollo and Mercury, which are both enterprise-level information management systems. In the future, we intend to explore federation at the tactical level.

Also, the Federation Service is being redesigned and integrated into the architecture for the next generation JBI – the Phoenix architecture. Phoenix is a services-based redesign of an information management system and will support both tactical environments as well as enterprise-level environments.

Further work is also planned for optimizing and improving the performance of federation and further evaluation, both in a laboratory setting as well as in the field.

References

1 Infospherics Web Site. Online reference: http://www.infospherics.org.

2 Linderman, M., et. al., "A Reference Model for Information Management to Support Coalition Information Sharing Needs", In Proceedings of 10th International Command and Control Research and Technology Symposium, 2005 http://www.dodccrp.org/events/10th_ICCRTS/CD/papers/274.pdf.

3 Marmelstein, R.E. "Force Templates – A Blueprint for Coalition Interaction within an Infosphere. In Proceedings of the Knowledge Systems for Coalition Operations Workshop (KSCO 2002). Toulouse, France. April, 2002.

4 Suri, N, Marcon, M., Quitadamo, R., Rebeschini, M., Arguedas, M., Stabellini, S., Tortonesi, M., Stefanelli, C. An Adaptive and Efficient Peer-to-Peer Service-oriented Architecture for MANET Environments with Agile Computing. In Proceedings of the Second IEEE Workshop on Autonomic Computing and Network Management (ACNM'08).

5 Carvalho, M., Suri, N., Arguedas, M., Rebeschini, M., and Breedy, M. A Cross-Layer Communications Framework for Tactical Environments. In Proceedings of the 2006 IEEE Military Communications Conference (MILCOM 2006), October 2006, Washington, D.C.

6 Tortonesi, M., Stefanelli, C., Suri, N., Arguedas, M., and Breedy, M. Mockets: A Novel Message-oriented Communication Middleware for the Wireless Internet, in *Proceedings of International Conference on Wireless Information Networks and Systems (WINSYS 2006)*, Setúbal, Portugal, August 2006.

7 Uszok, A., Bradshaw, J., Lott, J. Breedy, M., Bunch, L., Feltovich, P., Johnson, M. and Jung, H., (2008). New Developments in Ontology-Based Policy Management: Increasing the Practicality and Comprehensiveness of KAoS. In Proceedings of the IEEE Workshop on Policy 2008, IEEE Press.

8 Mitchell, G., Loyall, J., Webb, J., Gillen, M., Gronosky, A., and Atighetchi, M. A Software Architecture for Federating Information Spaces For Coalition Operations. In Proceedings of the 2008 Military Communications Conference (MilCom 2008), San Diego, CA, November 17-19, 2008.