Joint Battlespace Infosphere: Information Management within a C2 Enterprise

Vaughn T. Combs, Robert G. Hillman, Michael T. Muccio, Ryan W. McKeel
AFRL, Information Directorate, Systems & Information Interoperability Branch (IFSE)

ABSTRACT

System interoperability continues to be one of the key challenges facing the DoD as new capability is developed and fielded while maintaining operation with existing legacy systems. The concept of a Joint Battlespace Infosphere (JBI) was first created in 1999 by the US Air Force (USAF) Scientific Advisory Board (SAB) to address a clear lack of information engineering and interoperability within and among existing, traditionally stove-piped fielded capability.

Throughout the ensuing years, the Air Force Research Laboratory has performed research in information management that has further refined the original JBI concepts and has demonstrated how those concepts may be applied to achieve system interoperability. JBI information services allow for mediated, loosely-coupled access to information among systems and system components using a publish and subscribe paradigm. In addition, a query capability is provided for access to archived information. All data exchanged is treated as managed information. Each object published contains associated metadata that is used to broker for information object instances between producers and consumers. The consumers of information provide predicates or constraints over the metadata which are subsequently used to decide what information is appropriate for dissemination to the requesting clients.

More recently, DISA has sought to provide an architectural design and implementation of a collection of underlying services that would be available to edge user client applications and systems. These services allow clients to share information intelligently from anywhere within the network environment using a post, discover and pull methodology. These Net-Centric Enterprise Services (NCES) are being developed as a Service-Oriented Architecture (SOA) that uses the latest industry standards such as XML and Web Services.

This paper describes AFRL's internally developed JBI reference implementation version 1.2. The discussion will include information about the platform architecture. Future plans in the area of Infosphere research and development will also be discussed.

Finally, we discuss how the JBI core capability operating within the Global Information Grid (GIG) will utilize the NCES services to provide an information space capable of integrating, fusing, aggregating and intelligently disseminating information to warfighter business processes.

Keywords: Information, Interoperability, Information Management, Joint Battlespace Infosphere

1. The Joint Battlespace Infosphere (JBI)

The Joint Battlespace Infosphere (JBI) traces its origins to the USAF Scientific Advisory Board study that was completed in 1999 [1] [2]. In this document, they describe the vision of a combat information management system that would integrate information from a wide variety of sources, aggregate that information and then disseminate the information in the proper format and level of detail to appropriate consumers. In short, the JBI will provide the warfighter, subject to policy, access to the right information in the right format and at the right time.

An additional requirement of the JBI is to support a level of information interoperability that is absent in currently fielded systems. These systems tend to use point-to-point interfaces in order to link together traditionally stove-piped systems. To achieve the JBI interoperability goal, systems should be treated as a collection of loosely coupled components that issue requests for information that satisfy their individual requirements. This levies an additional derived requirement on the JBI and all of its clients that all information must be characterized before it is submitted to the Infosphere. It is in fact this metadata that plays a key role in the effective management and dissemination of information between producers and consumers. Information published to the Infosphere is often referred to as an Information Object (IO) or a Managed Information Object (MIO).

In order to consume information objects, clients may use one of two fundamental mechanisms: subscribe or query. Query is used to retrieve information that has been published to the JBI in the past, while subscription is used to receive information that may be published in the future. In both cases, a constraint or predicate is supplied over the metadata that is published with each MIO in order to assure that clients receive only the information that they are interested in.

The JBI provides developers with a capability for embedding legacy business logic or other transformation code as lightweight decision logic whose lifecycle is controlled by the underlying Infosphere. This decision logic, referred to as a *fuselet*, is completely maintained and managed by the JBI platform and may potentially act on behalf of a number of client applications.

In addition, the Infosphere provides a capability to dynamically alter the information space and the configuration of underlying services as operational units enter or leave the virtual infospace. Each deployed unit defines its capabilities, support needs, and information interface (subscription and publish descriptions) through the use of *Force Templates*. Force templates define the "electronic handshake" between the JBI and subordinate units, and their use lets units be quickly added to the JBI with little or no manual reconfiguration required. This capability allows individual units to identify their capabilities and needs via a collection of *Force Templates*.

Finally, the Infosphere may be dynamically administered by information Management Staff (IMS) personnel. It is their responsibility to enforce the Commander's intent represented as policy (security, QoS, etc.), which is enforced by the platform.

2. The AFRL JBI Reference Implementation

Soon after the SAB published their reports, engineers, and scientists in what is now the Systems and Information Interoperability Branch within the Air Force Research Laboratory (AFRL/IFSE) began researching the technologies and techniques that one could exploit to solve this rather unique set of information management challenges. As a product of this research several in-house Reference Implementations (RIs) were developed in order to prove efficacy of approach while providing developers and external researchers with a substrate upon which they could build their applications and research solutions. From the very beginning of the project, one philosophy reigned supreme over all others: pluggability. The intention was that researchers who would like to integrate there innovative solutions to some of the underlying information management challenges could do so using the existing RI's framework.

In this section we will provide a brief walk through the architecture of the most recently released RI (version 1.2).

2.1 Client Application

Before discussing the inner workings of the implementation it is appropriate to spend some time describing how client applications view and interact with the information management infrastructure. The application designer views the JBI infrastructure as a collection of information management services that will be used to maintain, disseminate, and otherwise manage information objects as they are published by producers and used by consumers.

2.1.1 The Information Object

The managed information object (IO) is the fundamental construct dealt with by clients and the Infosphere. At its simplest, it may be thought of as a container which is comprised of *metadata* that describes a particular information object instance and a *payload* which may contain the IO specific information. In addition, the IOs are each organized by type. The structure of the metadata that describes the IO (its *schema*) is defined and registered within the infosphere's Metadata Schema Repository (MSR) and associated with a type and version identifier. This subsystem provides a convenient set of interfaces that the client developers use when registering their types and that may be used when brokering for information. Since the published IOs are stamped with instance metadata that conforms to the structure and types stored in the MSR for that IO, a consuming client may issue a subscription *predicate* or query *constraint* that would match their exact information requirements. This increases the likelihood that the client will receive only the information that is of interest when it becomes available.

2.1.1 The Common API (CAPI)

During the development of several JBI implementations, a group of collaborating researchers (infospherics.org) developed a collection of interfaces that client applications would use when interacting with the Infosphere known as the JBI Common Application Programming Interface (CAPI). The infospherics group [3] is an external group consisting of government, commercial, and academic stakeholders. In short, the CAPI provides interfaces that describe the constructs that must be implemented by any compliant JBI implementation. This approach would allow for client application portability among a number of disparate JBI implementations. Five fundamental constructs exist within the CAPI, namely, the Connection Manager, Connection, Publisher Sequence, Subscriber Sequence, and Query Sequence. A client application first creates a Connection Manager which acts as a Connection factory. The Connection Manager is then used to create a Connection. The Connection then may be used to authenticate with the Infosphere in order to allow only authorized interactions by the client. The Connection, if authorized to do so, may then create one of a number of Publisher, Subscriber, or Query Sequences. Sequences are created based on the type of information object that the client either wishes to produce or consume. Sequences may also have attributes associated with them that are provided as requests to the platform that may or may not be honored based on existing policy. This notion was included to accommodate tailoring behaviors associated with a sequence in the future. One example would be a level of Quality of Service (QoS) requested by the client application. Another may be a requested level of resilience or fault tolerance. Subscriber and Query sequences may also provide a *predicate* or *constraint* defined over the metadata to further refine the definition of what is to be delivered to the consuming client. In addition, Subscriber Sequences may be used to specify a client implemented callback that provides for asynchronous delivery of information objects based on type and, optionally, matching predicate.

In keeping with the spirit of pluggabilty mentioned earlier, the Reference Implementation v1.2 provides for a completely pluggable CAPI implementation (depicted in Figure 1).

Currently the implementation class names (classes that implement the CAPI interfaces) are provided within a properties file. A platform developer need only provide the names of his CAPI implementation classes in order to allow our client runtime to interact with his platform.

The RI currently allows for two modes of underlying interaction with the platform. One mode allows interaction via a Web Services [4] layer using SOAP [5] messages. The second allows direct interaction with a collection of J2EE session beans. We used the pluggablity within the client-side implementation to choose between one mode and the other. In both cases the client uses the CAPI and is not aware of the underlying techniques and wire protocols that are being used.

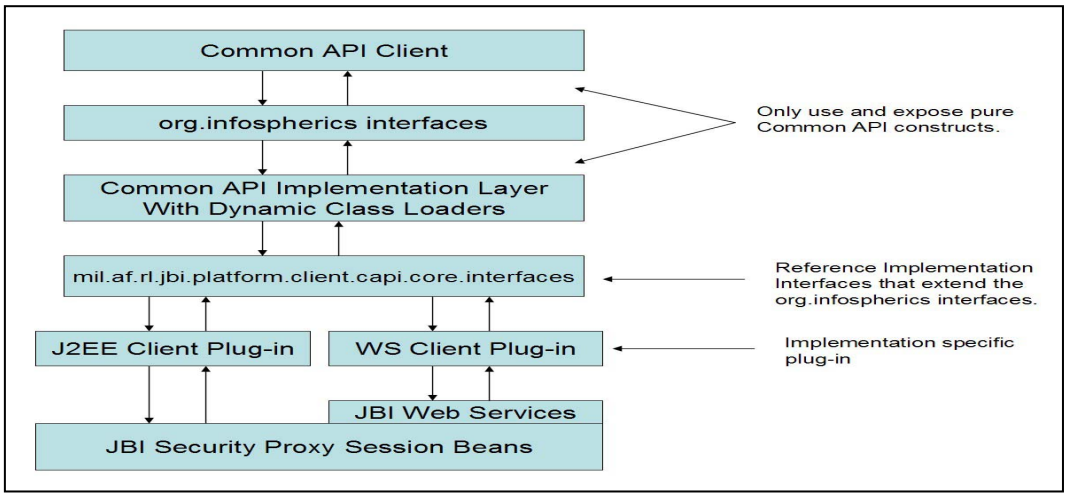

Figure 1: The Pluggable Common API

2.2 Version 1.2 Architectural Design

In this section we will provide a short description of the major services provided by the underlying information management infrastructure. A depiction of the RI version 1.2 architecture is provided. While this implementation could certainly be ported using a number of implementation technologies, we selected the Java 2 Enterprise Edition (J2EE) standard [6] for our current implementation. We selected JBoss [7] as an exemplar implementation of the J2EE standard. The JBoss application server is a robust and freely available implementation that is offered with source code. This makes it particularly attractive as a research vehicle.

We mentioned in the previous section that client applications may interact with the core services via two different modes. The first layer depicted in Figure 2 represents the server-side Web Services layer. The constructs available within the CAPI are implemented as a collection of services within the web tier that allow for interaction using the standard SOAP messages. The Web Services layer on the server-side is essentially nothing more than an alternate point of entry. It accepts SSL encrypted SOAP messages and processes the messages so that the appropriate Java object web service implementation is invoke with the proper parameters. The request is then handed off to the appropriate J2EE Security Proxy Stateless session layer. If the, more optimized, J2EE session mode of interaction is used by the client then the pluggable client implementation classes interact directly with the Security Proxy stateless session layer.

The Security Proxy Layer is used as the security gate through which all authorization checks are made. For example, as the client attempts to publish, subscribe, or query for a specific information object type, the security proxy sessions interact with the JBI Gatekeeper service in order to verify that the authenticated user currently has the sufficient level of privilege for the requested action. It is this role-based access control implementation that mediates all access to information and actions performed within the information management infrastructure.

Once the client interaction has been authorized, the request is passed to the generic implementation layer. This layer provides minimal additional processing and, for the most part, translates requests into concrete invocations on lower level implementation services. For example, if the request is from a publisher or subscriber requesting sequence activation then this would cause an interaction with components that provide for information object dissemination.

The InfoObject Dissemination Services or “Engine” has the primary role of providing a pluggable capability for predicate evaluation, subscriber queue management, and object proliferation or dissemination. Researchers may specify their own predicate evaluation implementations and drop them into the RI deployment. This allows for

ultimate flexibility as we explore different constraint languages and metadata representations. The implementation supports XPath [8] as a default constraint or predicate language. The predicate definition is an XML document

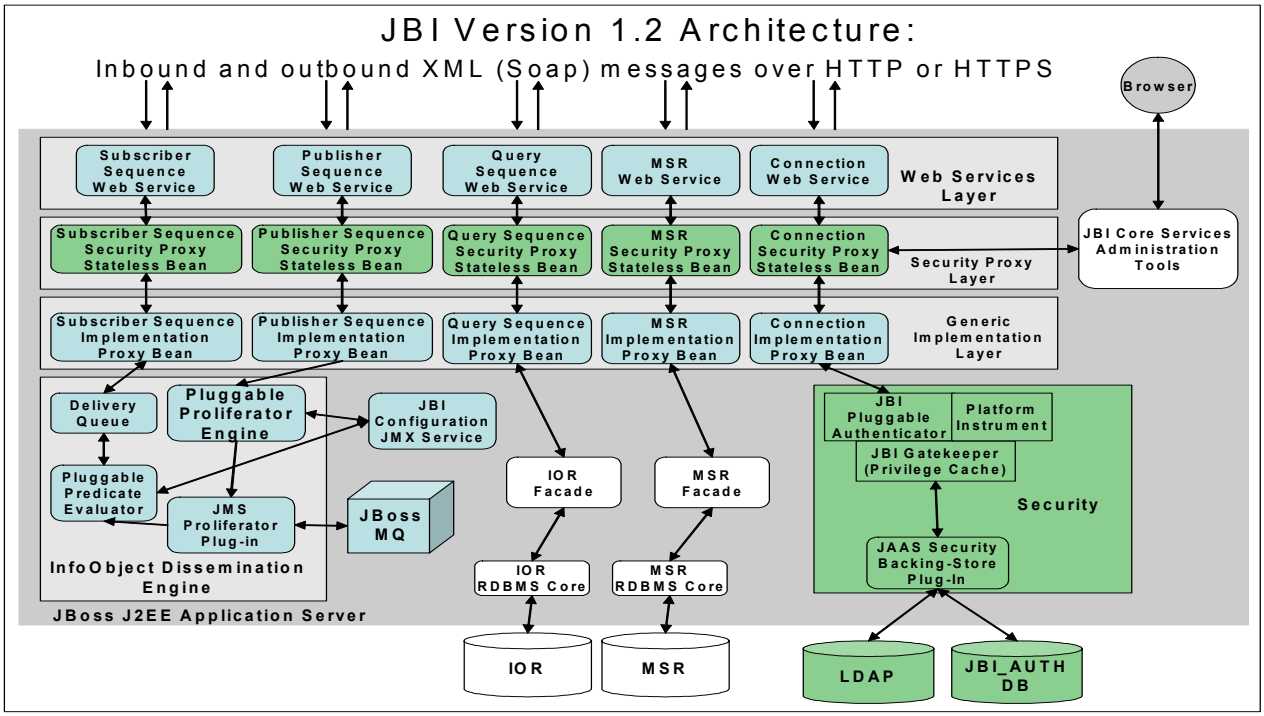

Figure 2: JBI Core Serivces

specifying a predicate name, predicate type, and the actual predicate. The client developer need only specify a predicate type that matches a particular predicate evaluator that is supported and the predicate will be associated and used for the specific subscription.

In addition, as a first attempt at providing a more flexible way of adding new wire protocols and dissemination technologies, we have introduced a *proliferator* and *receptor* model. The intent is that a proliferator and receptor be introduced and function as a pair. Normally the receptor would live within the consuming client's address space, while the proliferators, not surprisingly, would function within information management services layer (specifically within the InfoObject Dissemination Services). As an example, one of the default pairs that are used within the current version uses JMQ (a JBoss JMS implementation) to disseminate information objects to subscribing client consumers. A particular receptor may be specified within the aforementioned client properties file while matching proliferators may be developed and specified through the JBoss service console.

When an information object is published, its instance metadata is used in making a dissemination decision. The list of subscriptions associated with information objects of that type is formed and the appropriate pluggable predicate evaluator is used to produce the dissemination decision. At this point one of the dissemination services interacts with the JBI Gatekeeper to determine if the subscriber has the necessary privileges to receive the information object. If the object is to be disseminated, it is placed on the subscriber's queue. As the information object is pulled from the queue by the dissemination engine, the appropriate pluggable proliferator is used to push the object into the client subscriber's address space. It is at this point that the client's local callback is activated and the information object may be consumed.

When a query is issued via a Query Sequence, it is ultimately forwarded to the Information Object Repository (IOR) Façade Bean. This bean is located and used based on whatever underlying RDBMS implementation the substrate is using. In version 1.2, we support both MySQL and Oracle as underlying implementations where the XML metadata schema is mapped into relational tables in order for information objects to be retrieved based on selected the metadata elements. These data stores require a restart of the service when switching from one data store to another. In the future, this capability will be replaced by a JMX service (MBean), which allows for the hot data store swapping. In addition, future implementations will support XML database capability.

The query predicates that have bee defined in XPATH are converted to SQL for execution. Invocations are then forwarded to the appropriate underlying IOR core components that have the responsibility of issuing the appropriate SQL queries to the underlying database and returning information object result sets to the façade layer for delivery to client. It should also be mentioned that the Metadata Schema Repository (MSR) has client (CAPI) interfaces that allow for the specification of information object types and the metadata schemas associated with that type. Like all other interactions with the Infosphere, clients must have authorization to interact with the MSR. The underlying MSR Façade and Core component implementations mimic the IOR implementation closely.

The Security Services provide a JBI Pluggable Authenticator capability and a JBI Gatekeeper service. The Gatekeeper is the fundamental underlying service used in performing authorization checks and, in addition, provides for user privilege caching. The default authenticator uses a Java Authentication and Authorization Service (JAAS) [9] compliant backing store to maintain user privileges. When a specific Connection is authenticated the JBI Gatekeeper caches the privilege information obtained from the JAAS backing store and maintains it based on the connection UID. This approach is used in order to improve the efficiency of authorization checks.

A set of web based Information Management Staff (IMS) tools have also been developed, thereby easing the burden of administering the JBI. This set of interfaces allows for the creation of users (principals), association, and revocation of specific privileges with users, information object type and metadata schema definition, and information object repository management (object instance removal).

2.3 Future Directions

In this section, we will describe some of the new features and enhancements that will be coming in the next release of the JBI reference implementation and in future releases.

2.3.1 Discovery

The next release of the information management platform services will support two types of discovery. The client-side CAPI implementation classes are currently pluggable and dynamically loaded from libraries that are made available on the client's machine. In the next release, we will use a Universal Description, Discovery and Integration (UDDI) [10] server in order to discover an appropriate class server. This web service will be used to provide for the appropriate client-side class implementations based on information provided in the *connection descriptor*. The *connection descriptor* is currently provided when creating a new connection using the CAPI's ConnectionManager. It is currently underutilized but it will minimally identify the platform of interest by name and/or type. The class server will use this information in order to provide implementation classes that are compatible with a particular desired JBI platform implementation.

The UDDI server will also be used to obtain endpoint information that is necessary when directing invocations to a specific JBI service instance. This endpoint information would likely be useful regardless of the client and server-side implementation technologies that are being used. For example, in the current RI implementation, we provide two pluggable client-side CAPI implementations. One implementation uses Web Services, while the other interacts directly with J2EE session beans. In both cases, some endpoint information is necessary. In the Web Services case, a URI is necessary to direct invocations to the appropriate server-side Web Services CAPI implementation classes.

For the J2EE case, a Java Naming and Directory Interface (JNDI) [11] reference (a protocol specific URI) is used as the endpoint of interest. This will enable the discovered and dynamically loaded client-side CAPI implementation classes to interact seamlessly with appropriate information management services. We will further expand the notions of discovery to be of more generic use by implementing Infosphere system service components. This will allow for the more flexible use of components within the information management infrastructure.

In the future, we will be researching how mechanisms may be used in the discovery of *information* rather than implementing services. This is thought to be a longer term research project that may have implications on how we characterize and manage information within the existing Infosphere. As such, it probably will not appear in the next release of the reference implementation.

2.3.2 Policy Representation and Enforcement

The existing JBI implementation provides for a robust role based access control capability that allows for fairly fine-grained control over access to the Infosphere and dissemination of information. Within the RI, information object type names are specified within packages, similar to Java language implementations. Information management staff personnel may specify whether an authenticated user may Publish, Subscribe, Query, and/or Archive information objects of a specific type. In addition, wildcarding is allowed so that privileges may be defined for a package and all sub packages and types within the hierarchy. The language and mechanisms used to represent and enforce these access policies are inadequate and do not conform to a standard.

We are currently experimenting with different approaches for policy representation and enforcement within the Infosphere. As a first candidate, we are prototyping a role based access control capability using the eXtensible Access Control Markup Language (XACML) [12]. XACML is entirely XML based and was originally designed for access control. The language has been standardized by the OASIS (Organization for the Advancement of Structured Information Standards) group [13]. XACML embodies both an access control policy language (basically designed to describe who may do what and when) and a request-response based language that allows users to provide queries to determine whether or not a particular access should be allowed.

Typically a ***subject*** (principal or user in our parlance) may want to perform some ***action*** on a particular ***resource***. The subject would then issue a query on a service or component that has the responsibility for protecting the resource (in our case, information). This service is commonly referred to as the Policy Enforcement Point (PEP). The PEP service would then create a well formed and valid request based on the subject, action, and resource using the XACML request language. The PEP service would then forward the request to the Policy Decision Point (PDP) service or component which has the responsibility for evaluating the requests using applicable XACML policies in order to determine whether access should ultimately be granted. The response would then be returned back to the PEP service which either allows or denies access as appropriate.
XACML was selected as an initial target language for a number of reasons. It provides us with one standard access control policy language that would replace our rather limited representation. We envision using XACML in a more expansive role to represent and enforce policy throughout the information management system (i.e. QoS and resource management). We anticipate that the quantity and quality of tools for writing and managing XACML policies will be developed over time which will further alleviate some of the administrative burden placed on information management staff personnel. We have found that their policy language is more than adequate to represent all of the existing information access control policies that we have currently defined in our reference implementation. XACML will easily support access control policy specifications that are not currently available in the reference implementation (negative policy, etc.). The language is extensible and would seem to be able to accommodate all of the policy specifications that we may require in the future.

In addition, as part of an in-house research project, we are exploring how some of the more recently available semantic web languages may be used to represent and enforce policy within an information management system. Semantic web languages (DAML, OWL) have been researched with respect to policy representation in large, complex distributed systems [14]. The expressive capabilities of semantic web languages make them applicable to many domains. The available tools for manipulating, visualizing, and reasoning over these languages also provides

significant utility. One such application of semantic web languages to policy representation, reasoning, and enforcement is the Knowledgeable Agent-oriented System (KAoS) [15]. KAoS uses an ontological representation of policy and provides services that enable for the application, reasoning, and enforcement of these policies. KAoS also provides a powerful user interface KPAT (KAoS Policy AdministrationTool) that simplifies the authoring and manipulation of these policies. We are planning to explore how KAoS may be used within the JBI reference implementation in order to better describe policy based on the relationships that are defined between different types of information.

2.3.3 Instrumentation and Control

The JBI Instrumentation Services give users insight into what activity is occurring inside the reference implementation. The initial Instrumentation Services implementation was developed as a standalone system that interacted with the RI through a set of low impact interfaces. [16] This initial Instrumentation Services Architecture made use of the Instrumentation Entity Model to create entities that describe client objects interacting in the RI: platforms, connections, users, nodes, and sequences. These instrumentation entities (implemented with J2EE Entity Beans) populate the Instrumentation Space (a MySQL database) and are accessed by clients through the Instrumentation Client API (ICAPI). A web-based client that made use of this ICAPI was developed to visualize instrumentation information and demonstrate the capabilities of the Instrumentation Services. This client utilizes numerical rate graphing (using JFreeChart) and a dynamic graph tree (using TouchGraph) to visualize JBI activity, as seen in Figure 3.

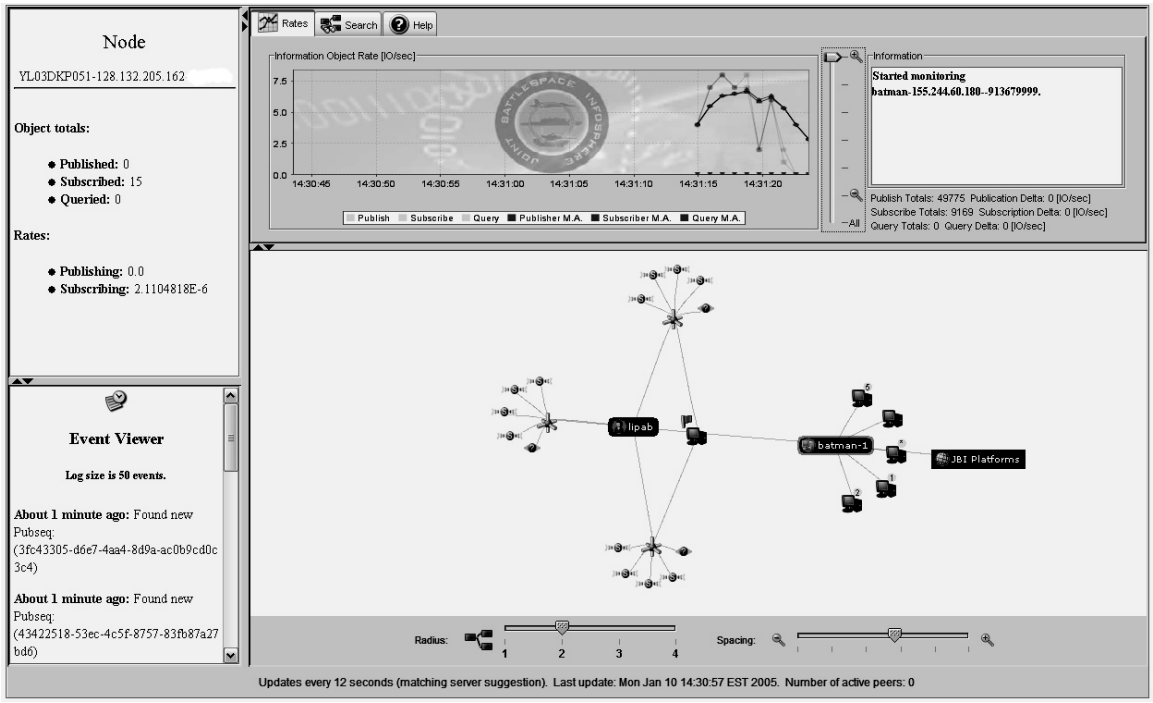

Figure 3: JBI Instrumentation Visualization Client

Current JBI Instrumentation Services development is focused on the Reference Implementation version 1.2.5 requirements. To make the Instrumentation Services Architecture more efficient, we have decided to move the Instrumentation Services into the RI. The RI now directly records instrumented events and information using JBoss Cache technology. Communication with instrumentation clients is still carried out through the ICAPI, which, in

version 1.2.5, wraps a local copy of the instrumentation cache maintained on the client by asynchronous messages transmitted from the RI. Added functionality to the Instrumentation Services will allow clients to display MIO traffic patterns and derive actionable statistics about MIO sources and destinations.

In addition to instrumentation, future ICAPI versions will provide instrumentation clients with control capabilities. Such control may include the ability to terminate connections, throttle back the MIO throughput of a sequence, or graphically modify user privileges. Combined with already existing instrumentation capabilities, this control functionality will allow clients to execute policy by monitoring instrumentation information, making policy decisions, and then effecting control over the RI. A client with this control capability would be an indispensable tool for the Information Management Staff. Current research into policy languages and engines will fuel ongoing instrumentation development.

2.3.4 Information Transformation

The transformation services will enhance the value of the information disseminated by the RI. The services will manage the storage, maintenance, and lifecycle of manipulation mechanisms known as fuselets. A fuselet is a lightweight, special-purpose client program that provides value-added information processing functions that are under the control of the RI. The fuselet's information processing functions manipulates incoming information objects in specific ways (defined by the fuselet logic) to produce new information objects for publication. In short, fuselets enable information to be manipulated into a form that is required by and useful to the warfighter.

The objective of fuselet technology is to enhance information systems by providing a flexible information production capability which may adapt to the changing needs of end users while requiring little or no change to legacy client applications. A fuselet runtime capability will have the effect of improving the efficiency and effectiveness of decision-making by correlating duplicative information, resolving inconsistent information, mediating between information sources, and fusing information together into comprehensible information products.

In order for fuselets to be useful their lifecycle must be carefully managed. The difficult task of developing of a runtime environment that manages a variety of fuselets implemented in multiple languages (XSLT, Jython, Java, Jess, Groovy, etc.), operate on a variety of information object types and payload formats, operate in multiple contexts (e.g., COI ontologies/semantics), and possess different execution properties (e.g., stateful versus stateless) is still underdevelopment. While the complete solution has yet been realized, AFRL has developed a prototype that establishes a basis for a fuselet execution and management environment [17]. The fuselet runtime environment (FRE) and fuselet runtime management environment (FRME) are currently being developed and will be deployed a future release of the reference implementation.

Section 3: JBI and Net-Centric Synergy

A key challenge for the DoD is to "improve the speed and quality of [warfighter] decision making by connecting information producers and consumers more effectively through information technology and net-centricity." The JBI project is experimenting with advanced information management technologies and applying those concepts in the formulation and evolution of an effectively managed information space. We anticipate a variety of domain–specific information spaces being identified to support groups of users exchanging information in pursuit of shared goals or missions. These groups of users are referred to as a Communities of Interest (COI).

3.1 Community of Interest

COI [18] is a term used to describe any collaborative group of users who must exchange information in pursuit of their shared goals, interests, missions, or business processes, and who therefore must have shared vocabulary for the information they exchange. The COI concept is intended to be broad, and cover an enormous number of potential

groups of every kind and size. In the creation of a COI there are several tasks that must be completed before an effective net-centric sharing process can be achieved. First, a detailed information engineering process must be performed to develop the information object types and metadata schemas that will be used as the vocabulary to provide the semantic and syntactic understanding within the share information space. The metadata schema should then be registered in the DoD Metadata Registry for visibility and reuse. This will allow other communities and users throughout the DoD to use related constructs to exchange information. Reusing existing metadata will tend to require less mediation or transformation when information is exchanged between COI's.

3.1 Air Force C2 Constellation

The C2 Constellation [19] program has been created to assist the Air Force in building a network centric, peer based, system of systems, which operate in a seamless and fully interoperable framework. The Air Force vision is a connected array of land, platform, and spaced based sensors that use common standards and communication protocols to relay information automatically in what he refers to as "machine to machine interface." This effort to integrate systems is referred to as enterprise integration. The C2 Constellation is tackling the networking solutions and connectivity issues in a two-tiered approach. At the top tier the C2 Constellation seeks to define C4ISR enterprise integration. At the bottom tier the C2 Constellation solves near term, quick turnaround integration solutions for the Air Force. The JBI research team is evaluating the C2 architecture and how the JBI IM services can be can support edge user development and integration. The current COI application communities within the C4ISR enterprise being targeted by the JBI research team are the intelligence and tactical communities, with additional feasibility demonstrations in the C2 community.

3.2 Global Information Grid Enterprise Services

Several initiatives have been undertaken to establish net-centric capabilities across the DoD. The Global Information Grid (GIG) [20] will be a net-centric system operating in a global context. The system will provide processing, storage, management, and transport of information. The GIG will support the Department of Defense (DoD), national security agencies, and related Intelligence Community missions and functions across strategic, operational, tactical, and business-in war, and will do so in crisis, as well as peace. The GIG will provide the foundation for net-centric operations by globally interconnecting the building blocks of information capabilities, including: the physical communication links, hardware, software, data, personnel and processes. The GIG Enterprise Services (ES) is an umbrella term to describe the information services that reside on the GIG. The Network Centric Enterprise Services (NCES) is a DISA-sponsored program that seeks to provide a generic set of core services that will be available to all DoD edge users allowing them to "pull mission-tailored information intelligently from anywhere within the network environment [21]."

3.3 Net-Centric Enterprise Services

The initial NCES core services are being developed using the latest industry standards, such as extensible markup language (XML) and web services, and will provide functionality as services to GIG end-user applications. This architectural approach and its use of standards and web technologies are referred to as a Service-Oriented Architecture (SOA). SOA and other enterprise architecture approaches are based on a collection of independent and networked business components where all interactions are achieved through well-defined, published, and standards-compliant interfaces. The SOA concept defines three levels of abstraction: Operations, Services, and Business Processes. Operations represent single units of work, and when logically grouped, form a Service. The table below illustrates an example of a service called "Inventory Control" consisting of several operations related to inventory items.

Service	***Operations***
Inventory Control	Lookup items by reference number
	List resources by name and region
	Save data for new items

A business process consists of a long running set of actions or activities performed with specific business goals in mind, and when layered on top of work order processing, would involve activities such as "Sell Products" or "Fulfill Orders". Business processes typically encompass multiple service invocations either one at a time or several at a time in an orchestrated fashion [22]. Examples of USAF business processes could be activities conducted in support of an air campaign, such as Air Tasking Order (ATO) planning or Target Nomination List (TNL) generation.

3.4 JBI and GIG Enterprise Services

Within the GIG infrastructure, multiple JBI-enabled COIs and other edge user applications will be established to satisfy specific end user needs. These JBI-enabled COIs will be capable of sharing their information among each other as well as other edge users operating on the GIG. The net result is that each user will have capabilities above and beyond what any individual service capability provides.

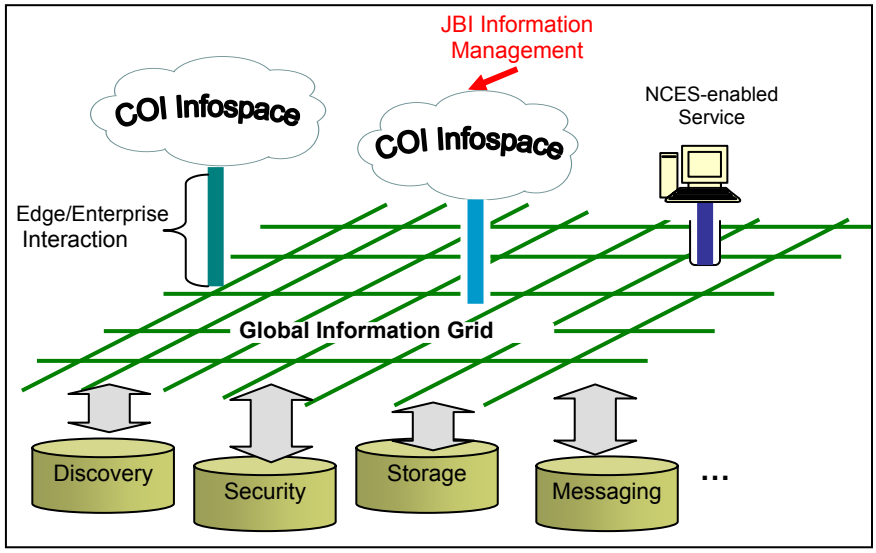

Figure 4: JBI enabled Information space facilitate edge user interaction in conjunction with GIG enterprise services.

Figure 4 illustrates this overall concept of NCES- and JBI-enabled COIs operating in conjunction with other edge user applications existing on top of the underlying GIG foundation. In reality, the GIG infrastructure will provide the DoD enterprise transparent and seamless access between the NCES core services, JBI information spaces and edge user services. On their own, JBI Information Management (IM) services provide information management and exchange capabilities that support tailored, dynamic, and timely access to required information to enable near real-time planning, control, and execution for DoD decision making. However, JBI IM services will be capable of integrating with the NCES core services and when deployed, will establish an interoperable "information space" that aggregates, integrates, fuses, and intelligently disseminates relevant information to support effective warfighter business processes. This virtual information space provides individual users with information tailored to their specific functional responsibilities and provides a highly tailored repository of, or access to, information that is designed to support a specific COI, geographic area or mission. This information space also serves as a clearinghouse and a workspace for anyone contributing to the accomplishment of the operation—for example, weather, intelligence, logistics, or other support personnel.

In combination with NCES core services, JBI IM services will be utilized to dynamically create an information space and manage edge user interaction within a COI. An example of this, from a GIG ES perspective, would allow

edge users to discover relevant COIs (using NCES discover services), gain access to the information space (JBI access controls using NCES user profiles and underlying security mechanisms), and exchange or manipulate relevant information (using JBI IM services). In this manner, a JBI-enabled COI would have the full suite of JBI information management capabilities, but would still be based on and have full access to NCES services. Together, NCES and JBI IM services will be utilized to develop end user applications targeted at mission-specific COIs to facilitate a secure, controlled information exchange and decision support environment.

3.5 JBI and GIG Net-Centric Services

The JBI research team has established a local NCES testbed, a Solaris server which hosts the current NCES development tree and available services. The intent is to host the emerging NCES services as they are being developing and evaluate their application. The goal is to identify JBI, GIG-ES, and C2 Constellation gaps and intersections. The JBI research team will take the opportunity to influence potential areas where JBI provides a capability that does not exist within the GIG ES. It is critical that the available services are evaluated to identify which ones can be directly used, how the JBI IM services can be integrated within the GIG ES and what potential technical hurdles must be overcome to provide a cohesive foundation for edge user development and integration. Figure 5 illustrates an initial evaluation that was conducted by one of the JBI research contractors as a starting point in assessing potential JBI and GIG ES synergy. The results of this analysis will be pperented as a separate publication.

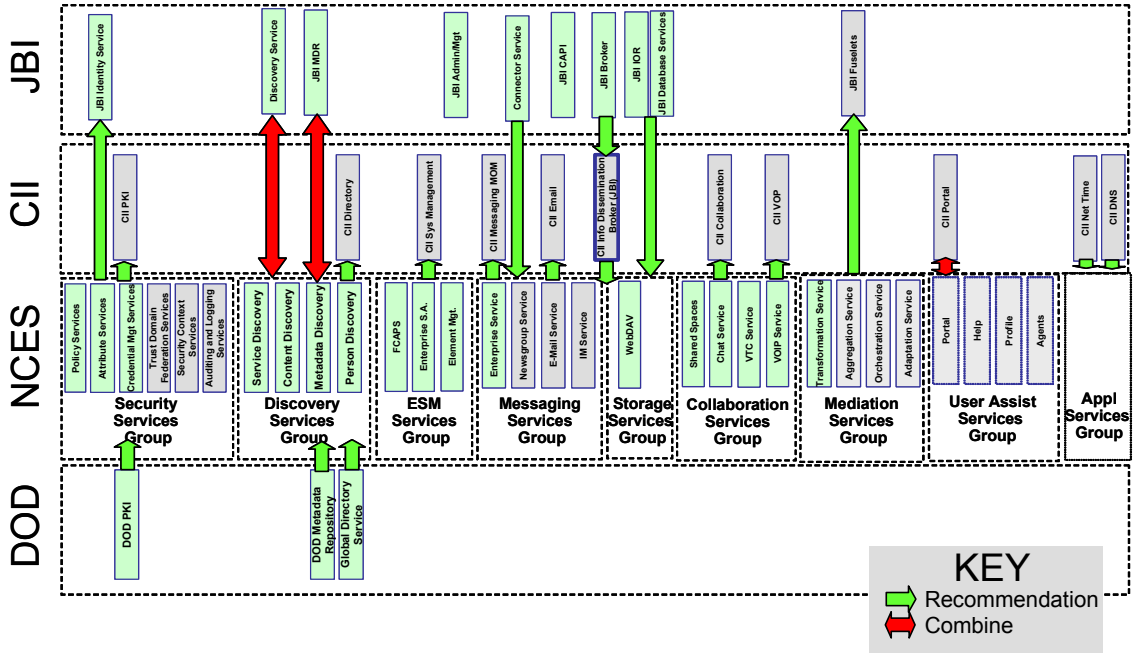

Figure 5: Initial Evaluation of GIG ES intersection

Section 4: Conclusion

Current research and development under the JBI information management technology area is focused on a vision of effectively building and managing an information space. The JBI Common API provides an isolation layer between client applications and the underlying technology. This technology isolation allows this reference implementation or any implementation developed to be based on the latest commercial technologies and therefore insure that future innovation can continue as the commercial IT technology base moves forward. The JBI enabled information space

is intended to support virtually any operational domain; short term utilization such as humanitarian relief to larger long-term COIs supporting edge users interacting within the GIG. Future work will address the information engineering process to support domain-specific COI interactions including internal legacy systems as well as the transformation requirements to enable interoperability among various COIs residing on the GIG.

REFERENCES

1. Scientific Advisory Board, " Building the Joint Battlespace Infosphere Volume 1: Summary," SAB-TR-99-02, 2000.
2. Scientific Advisory Board, "Report on Building the Joint Battlespace Infosphere Volume 2: Interactive Information Technologies," SAB-TR-99-02, 1999.
3. The Infospherics Group, http://www.infospherics.org
4. W3C, "Web Services," http://www.w3.org/2002/ws/, 2002.
5. W3C, "SOAP," http://www.w3.org/TR/soap/, 2003.
6. Sun Microsystems, "J2EE Version 1.3 Specification," http://java.sun.com/j2ee/j2ee-1_3-fr-spec.pdf, 2001.
7. JBoss, "JBoss Application Server," http://www.jboss.org/products/jbossas, 2005.
8. W3C, "XPath," http://www.w3.org/TR/2005/WD-xpath20-20050211/, 2005.
9. Sun Microsystems, "Java Authentication and Authorization Service (JAAS)," http://java.sun.com/products/jaas/, 2004.
10. OASIS, "UDDI Specification," http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm#uddiv3, 2004.
11. Sun Microsystems, "Java Naming and Directory Interface (JNDI)," http://java.sun.com/products/jndi/, 1999.
12. OASIS, "eXtensible Access Control Markup Language (XACML)," http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml, 2005.
13. OASIS, http://www.oasis-open.org/home/index.php, 1993.
14. G. Tonti et al., "Semantic Web Languages for Policy Representation and Reasoning: A Comparison of KAoS, Rei, and Ponder," The Semantic Web-ISWC 2003: 2nd International Semantic Web Conference, LNCS 2870, Springer-Verlag, 2003, pp. 419-437.
15. Andrzej Uszok et al., "KAoS Policy Management for Semantic Web Services," IEEE Intelligent Systems, July 2004, pp. 32-41.
16. M. T. Muccio et al., "JBI health and status instrumentation services," SPIE Defense & Security Symposium, March 2005
17. N. O. Ahmed et al., "Fuselets: lightweight applications for information manipulation," SPIE Defense & Security Symposium, March 2005
18. The Department of Defense, Chief Information Officer, Information Management Directorate,"Communities of Interest in the Net-Centric DoD Frequently Asked Questions (FAQ)," May 19, 2004
19. Col Normal Sweet et al.,"The C2 Constellation A US Air Force Network Centric Warfare Program", CRRTS, June 2004
20. Global Information Grid, http://www.nsa.gov/ia/industry/gig.cfm
21. Department of Defense web page, Global Information Grid Enterprise Services The Challenge, https://ges.dod.mil/about/challenge.htm, August 20, 2004
22. Olaf Zimmermann, Pal Krogdahl, Clive Gee, "Elements of Service-Oriented Analysis and Design" http://www-106.ibm.com/developerworks/webservices/library/ws-soad1/, August 20, 2004