

# Architecture of the **FHIE**

How the DoD and the VA Are Integrating 10 Million Health Records

An ambitious project offers lessons for the NHIN and illustrates core HIM issues in data exchange.

In response to the reported illnesses of Gulf War veterans, the Department of Defense (DoD) and Department of Veterans Affairs (VA) undertook an initiative to create comprehensive, lifelong medical records for every military personnel. That program was called the Government Computer-based Patient Record project, now renamed the Federal Health Information Exchange (FHIE).

FHIE integrates the lifelong records of more than 10 million active duty and retired members of the US armed forces. It includes substantial infrastructure for exchanging electronic health records between the VA and DoD Military Health System. As of April 2006, 11 DoD Composite Health Care System 1 host systems supporting 11 DoD Medical Center Systems, seven DoD hospitals, and more than 100 DoD clinics were added to the VA network, which already connects about 150 Veterans Health Administration VistA patient record systems.

Implemented in 2002, the FHIE architecture has been used as a near-term solution, moving records for separated military personnel (those retired or discharged) from the DoD to the Veterans Health Administration. In 2004 it became the basis for the Bidirectional Health Information Exchange (BHIE), which exchanges records both ways between existing healthcare systems. In 2006 BHIE was awarded an Excellence Gov award by the American Council for Technology.

The FHIE system architecture integrates healthcare IT systems to identify patients and to exchange administrative and clinical records. The architecture is described here from four viewpoints: system connection, framework, information, and deployment.

The experience of an integration on this scale provides useful lessons for the evolution of the nationwide health information network now under consideration.

by **Greg W. Donham**  
and **Tony Mallia**

## System Connection Architecture

The FHIE architecture supports two types of services—person services and record retrieval services. Person service is the identification and locator service; record retrieval is the fetching and aggregation service. Multiple instances of these services exist in domains across the framework.

A domain is a community of patients. It is the basis for managing patient identity, privacy, and terminology.

The approach, which is well established in enterprise integration projects, intermediates between the data structures in the connected systems (standard or proprietary) using a normalized, canonical form to which all incoming records and from which all outgoing records are transformed. This canonical form is described below under information architecture. The effect of the approach is to minimize the cost of the addition of a new system with a new variant of a standard.

As an example, it was possible to design and implement the IHE XDS protocol (Integrating the Health Enterprise Cross Enterprise Document Sharing) and HL7 profiles in about 90 days to demonstration quality.

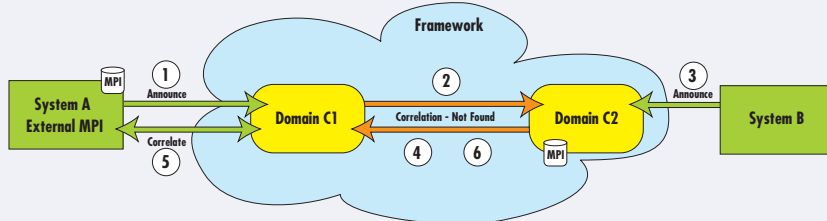
Integration with existing healthcare systems is perhaps the most challenging aspect of the architecture. The principles of the architecture were to adopt as flexible an approach as possible to support both the record-consuming (requesting) and record-providing systems.

### Person Identification

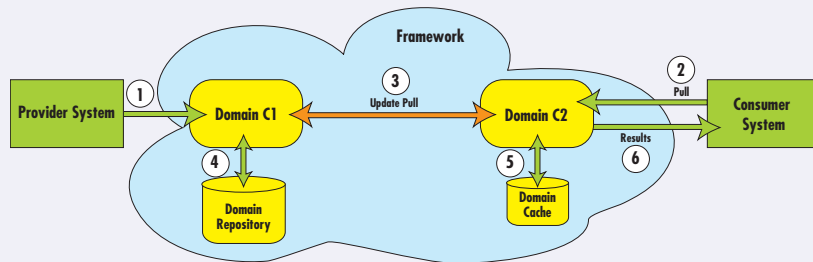
In order to avoid a central person repository, patient identification builds on the identification management processes of existing healthcare systems and correlates the identities across the framework. Either matching algorithms of the existing healthcare systems or matching logic in the framework can be implemented.

As illustrated in the figure “Person Identification,” above, system A announces the patient identity (1), and the framework attempts to correlate with other domains and fails (2). System B announces the same patient (3) and requests the correlation (4) with other domains. Domain C1 correlates against system A with its master person index (MPI) (5) and provides the corresponding identity (6), which is recorded in domain C2.

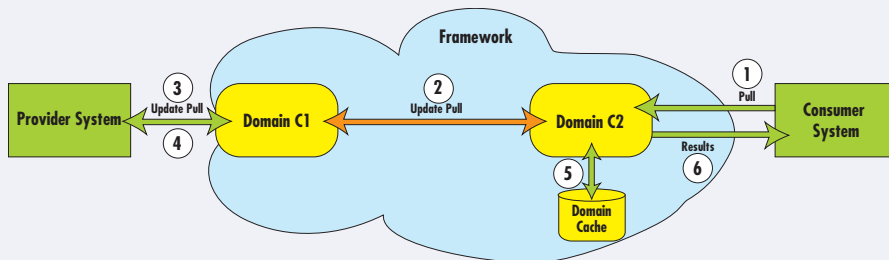
### Person Identification



### Push-Pull Scheme



### Pull Scheme



Domains configured with internal or external MPIs may be mixed inside the framework.

### Record Retrieval Service

In anticipation of multiple searches for a patient during an encounter or care episode, the records retrieved by a previous request are stored in a cache in the domain of the retrieving system so that future searches will only fetch updates from the providing systems.

The destination cache has an algorithm for determining how long it can remain before its information is no longer current. This value can be set to zero to force an update for every query.

At present, the consuming client is assumed to pull, or request, the records for the patient. (An asynchronous push, or notify, mechanism is listed in future directions.) In turn, the framework will support either a pull from the providing system or a previous push—in which case an intermediate repository then supports the pull.

In the push-pull scheme (see above), the first event (1) is the provider system notifying or pushing the record to a repository. At a later time the consumer system makes a pull request (2) for the patient’s records. The update to the cache is requested (3) from all domains at which the patient is known. Domain C1 pulls the records from the repository (4) and returns it to the cache (5), where the total results are sorted in reverse chronological sequence and passed to the consuming system (6).

In the pull scheme (see above), the sequence is the same as the push-pull scheme except that the record is pulled from the provider system directly (4) instead of the repository.

Only one scheme should be used for a provider system to avoid generating duplicate records. However, schemes can be intermixed across multiple provider systems within the same domain, and the configuration of one domain is independent from another.

## Framework Architecture

The framework architecture is a set of interconnected domains. The domain is therefore a fundamental building block in the application and deployment architectures.

Each specific domain, or domain instance, has relationships to a parent or child or both. A domain is connected to its immediate parent or children but not to its siblings (see “Domain Relationships,” right).

The architecture is closely aligned to service-oriented architecture, common in enterprise integration.

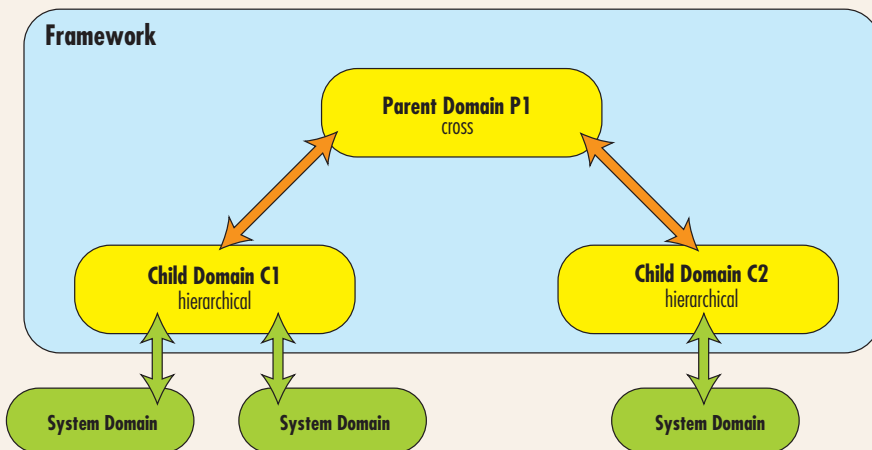
So far, a domain has been described as a community of patient information. It is also a unit of software deployment that groups the software components. Inside a domain there are services and adapters, which may be configured for a particular domain instance. (See “Inside the Domain.”)

Adapters translate between the domain and external systems. Provider adapters (1) support client access into the domain, and consumer adapters (2) support client access out of the domain.

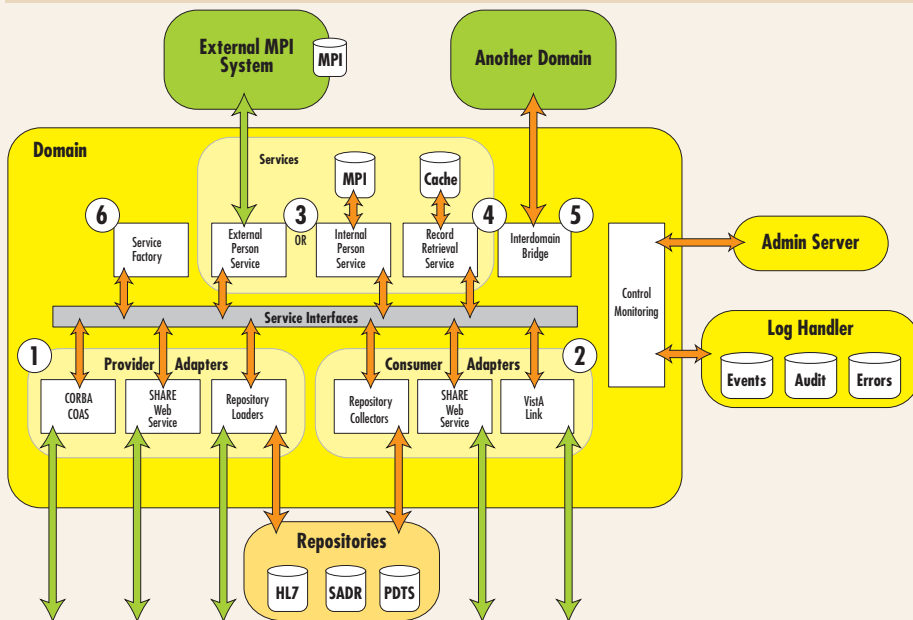
The configuration of the domain, including its instance identity and the association with its parent and children, is done through property descriptors. The software code for each domain is identical.

The core framework includes person services (3) and record retrieval services (4), which are invoked from adapters and will invoke other services or outgoing adapters. A service may be constructed and invoked in the same domain or in an immediate parent or child domain.

### Domain Relationships



### Inside the Domain



### Distributed Connected Domains

The interdomain bridge (5) allows the domains to be locally or remotely located. The service factory (6) provides the requesting client access to the requested service either directly if it is the same server or through a proxy if it is remote. The client is unaware of the difference and is unaware of any middleware used to communicate with the remote domain.

### Service Framework

Information is moved across the service framework and over the interdomain bridge in arrays of one or more object graphs (described under information architecture, below).

All services and consumer adapters implement an interface that supports the behavior of a virtual collection of graphs. It assumes that the service represents a collection of people or records and that the functions will add a graph to the collection, get a graph from a collection matching a partial graph submitted, and get selected graphs from the collection based on a selector. There are also special operations such as merge, unmerge, transactions commit, and rollback.

Services are assumed to be stateful and persist after the service factory has created them until the “done” operation.

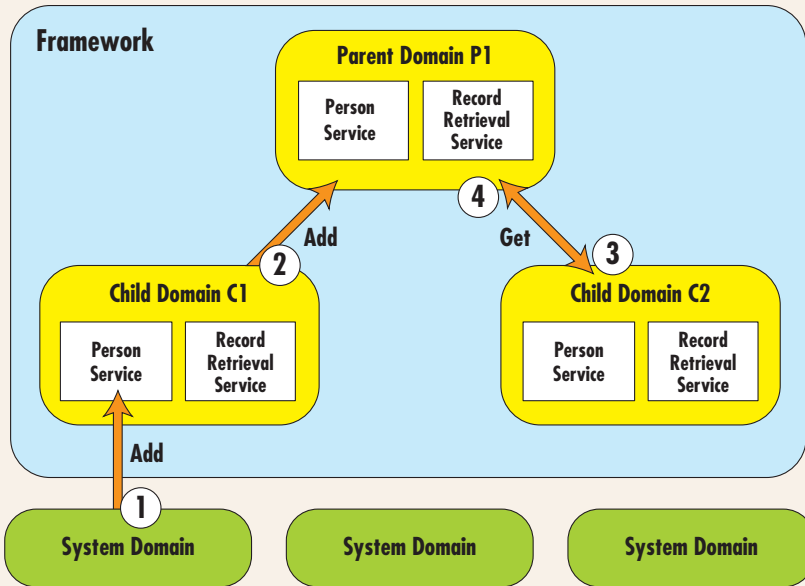
### Person Service

The framework is derived from the OMG PIDS correlating ID domain concept, which is referred to as a hierarchical domain since it contains the union or superset of all patients in the lower or child domains.

However, since a hierarchical model would inevitably result in a domain with everyone in it, another type of correlating ID domain is used, called a cross domain. A cross domain contains the intersection of its children’s patient populations.

Each domain can be configured as a hierarchical or cross-correlating domain. The lowest child domains in the frame-

**Patient Matching**



patient identity demographics (4). If the patient is found in two or more child domains, the identity is stored in the parent (because it is a cross domain).

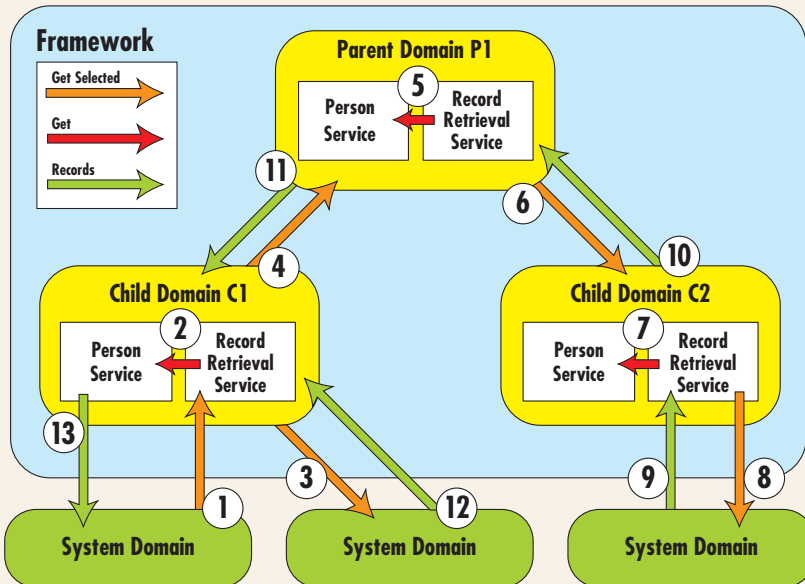
**Record Retrieval Service**

The record retrieval service has a more complicated pattern.

As illustrated in the figure “Record Retrieval,” the client system requests records for a patient using its patient identity with a get selected operation (1). The record retrieval service pulls the other identities from the person service (2) and queries its cache for currency. If the cache is not current, it issues the query for the update records to it (3) and to the parent domain using its correlated ID (4).

Using the local identifier, the cross domain checks if it knows the patient and, if it does, what other child domains know the patient and by what identifier (5). It then issues a get selected command with the patient ID for that domain (6). That domain’s record retrieval service finds the local identity from its person service (7) and, using that identity, queries the local system for the records (8). The records are returned (9) and passed back to the cross domain (10) and then to the calling domain (11). At this point, as would happen at any other domain if there had been multiple respondents, the records are retrieved from the local system (12) and merged with the records from the cross domain (11). The cache is updated, and the whole set is sorted into reverse chronological sequence and passed back to the original querying system (13).

**Record Retrieval**



work (C1 and C2 in the earlier figure) must be hierarchical, since they must know about all the patients in the connected system domains. Normally the highest domain (P1 in the figure) would be configured as a cross domain.

The person service performs two primary functions—the correlation of patient identities and the retrieval of child domain identities for record location. It does this with an add operation to announce a patient identity and a get operation

to find the matching patient.

In the figure “Patient Matching,” above, the local system announces the patient with an add to the attached domain’s person service (1). If the person service does not already know the patient, it announces the patient to its parent (2). If the parent domain does not know the patient, it asks its other child domain(s) to match the patient with a get operation (3) and, if found (regardless of whether it is an internal or external MPI), will return the

**Provider Adapters**

Provider adapters offer the framework services to outside consumers. They convert the query and notification operations of the appropriate protocol into the internal add and get selected operations of the framework. In addition, the adapters transform the structure of the messages to and from the canonical form used internally.

- ▶ CORBA COAS communicates between the framework and the Veterans Health Administration (VHA) boundary system for VHA queries. The boundary system is a gateway

**Framework Architecture *continued***

that interacts with all the other VHA systems.

- ▶ SHARE Web service is used for the DoD's Composite Health Care System 1 systems to make queries to the framework.
- ▶ External person service is used for the VHA MPI to notify the framework of new patients.
- ▶ Repository loaders are used for the various feeds to notify the framework of new records.

**Consumer Adapters**

Consumer adapters support the domain services consuming services outside the domain. They are separated into repository collectors and interactive collectors.

Repository collectors implement the get

selected operation with a database query to the repositories and retrieve and transform records to the canonical form.

- ▶ Health Level Seven (HL7) enables collectors for both HL7 2.2 DoD profile as well as for HL7 2.4.
- ▶ PDTS collector pulls records from the DoD Pharmacy Data Transaction Service, formatted under the standard defined by the National Council for Prescription Drug Programs.
- ▶ SADR pulls records from the DoD Standard Ambulatory Data Record repository.

Interactive collectors convert the get selected operation to that operation in the protocol, which will retrieve the records from the target system and transform

them to the canonical form.

- ▶ SHARE Web service collects records from the DoD's Composite Health Care System 1 systems.
- ▶ VistALink collects records from the VistA Computerized Patient Record Systems through the VHA boundary system.

The interdomain bridge adapter provides communication for the service factory, person service, and record retrieval service to communicate with their adjacent domain services. It provides the service operations over CORBA but isolates the primary services from knowledge of this transport. The service factory is the only component that is aware of the bridge.

**Information Architecture**

The information architecture relies on the canonical form in which patient identity demographics and health records are passed within and between the domains and stored in the cache.

**Repository Formats**

Repositories on the other hand, store information in the form that the records have been received (e.g., HL7), and the collector, not the loader, is responsible for transforming the record into the required canonical form. This allows change of transformation rules and new versions of the canonical form to be introduced without having to rebuild the repositories.

**Information Model**

The canonical information structure is defined by a UML Information Model called the VHA Health Information Model (VHIM). FHIE implements VHIM version 2.1.

The lowest level of concept is called a datatype; examples include CodedElement, DateTime, NumericValue, Plain-

Text, and TimeSpan. Although these datatypes have internal structure, they are considered to be atomic and thus can't be shared or associated with other elements. Images, such as x-rays, will be added during 2006.

The bulk of the VHIM 2.1 is composed of eight packages with multiple versions covering laboratory, material, medications, patient encounters, person demographics, text documents, vital signs, and working lists. It has been aligned mainly with HL7 v2.

**Templates**

In order to define the canonical form for records within the system, a view that is a restriction of the elements in the information model is used to establish the record template. These scope definitions are called templates. The graph that is formed is effectively a tree structure from the root outward to the datatypes.

Each template is named with a specific version as part of the name. Therefore multiple template versions can coexist

in the model and can coexist side by side in the framework, because the template name is one of the parameters of the get selected operation.

The structures can change without recompilation of any system components because the formation of the graphs by an adapter is done according to mapping files, which are produced as part of the development process and deployed into the domain server while running.

Templates, and therefore internal canonical forms, have been deployed for:

- ▶ Consults
- ▶ Laboratory results, including anatomic pathology, chemistry, and microbiology
- ▶ Outpatient medication orders and dispensing
- ▶ Patient encounters, including discharge summary
- ▶ Progress notes, including pre- and post-deployment health assessments
- ▶ Radiology text reports
- ▶ Vital signs
- ▶ Working lists, including allergies

## Deployment Architecture

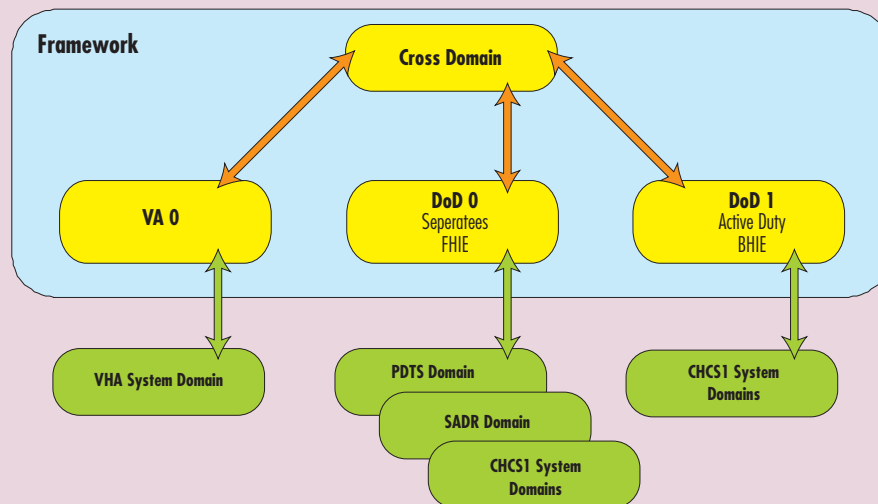
The actual FHIE and BHIE deployment as of January 2006 has four domains:

- ▶ The FHIE cross domain
- ▶ VA 0: the VHA
- ▶ DoD 0: the military separatees
- ▶ DoD 1: the active duty members

Plans call for a consolidation of the DoD domains in the future.

Work is under way to relocate the domains so that the interdomain bridge communication should be the only mechanism crossing the firewalls. This was the original intent of the architecture.

**FHIE Domains, January 2006**



## Future Deployment Directions and Lessons Learned

A number of architectural issues are under investigation as part of the framework's evolution:

- ▶ Peer-to-peer domains to avoid the cross-correlating domain. Multiple organizations that collaborate do not have a location to deploy a shared server.
- ▶ Extended asynchronous behavior. As more organizations join the framework, the performance of certain sequences such as patient correlation need to be done in a more decoupled, independent fashion.
- ▶ New platforms are emerging, including the enterprise service bus, with standards that lend themselves to the framework.
- ▶ Graph representation and the adoption of standards. The internal representation of data could move toward an industry standard, which would provide more common support routines.

The architecture appears to work well. System performance is very robust, with user availability ("uptime") at more than 99 percent. Integration of new adapters is quite fast. The time to implement the first production integration for FHIE in 2002 took 10 months to get the first iteration system to test.

A number of functions were found to work well:

- ▶ The canonical information format with multiple side-by-side versions
- ▶ The collaboration between the services over the service framework

The functions requiring more attention than expected were:

- ▶ Patient identity matching and the treatment of false negatives
- ▶ Clinical record identity and the detection and removal of duplicates
- ▶ Validation of upstream system data quality

Some functions have not performed as expected and deserve some rethinking:

- ▶ The cache does not always give the response improvement expected because the loading algorithm is not optimized for the observed pattern of repeat queries.
- ▶ Quality levels need to be established for external specifications and protocols so that there is no ambiguity in the options that may be implemented.

Finally the most important lesson learned was not to base an integration architecture solely on systems that are yet to be developed. Even if the new system is ready, there is a significant period of time during which there is parallel operation and migration between the old and the new.

The first step should be to service-enable the legacy systems, allowing them to act as service providers to other systems. ❖

## Reference

Intergovernmental Advisory Board, American Council for Technology. "Health IT in Government: Transforming Healthcare and Empowering Citizens." March 2006. Available online at <http://colab.cim3.net/file/work/IAB/HIT%20in%20Govt%20Report%203.06.pdf>.

## Acknowledgments

The authors wish to acknowledge the significant contributions from Robert M. Kolodner, MD, and James Demetriades of the Department of Veterans Affairs. Also, Philip C. Dederer, RHIA, in Military Health Systems and James Reardon for his executive vision and support of FHIE and BHIE while he was the Military Health System CIO at the Department of Defense. Finally, to the countless numbers of US government and private sector staff who began this journey as part of the Government Computer-based Patient Record system in 1998.

**Greg W. Donham** is the interagency program director for BHIE/FHIE at the Office of Information/Health Enterprise Strategy at the Department of Veterans Affairs. **Tony Mallia** ([amallia@ciber.com](mailto:amallia@ciber.com)) is a principal consultant with CIBER, Inc., and is responsible for the architecture for FHIE at the Department of Veterans Affairs.