# Combined Navy Developer's Guide

# DRAFT – 4/25/02

# Table of Contents

# 1.O Introduction

There is no shortage of historical writings that chronicle the evolution of the Navy and Marine Corps since their inception. During the close of the 20th Century we focused on the Revolution in Military Affairs and all that it portended. We saw a beginning of the shifts in warfare doctrine to the current approaches to Network Centric Warfare. The Navy today is pursuing the digital network in support of its Information Technology for the 21st Century (IT-21) and the Marines have established both a Marine Corps Enterprise Network (MCEN) and a Marine Corps Tactical Network (MCTN). We are a global military force that requires a global network—one that seamlessly links forces at sea with support bases and technologies ashore. Networks have become key to current and future Naval war fighting. A Global network will also allow the establishment of common databases and lead to the use of standard approaches to problem solving. The Naval Forces requires this kind of network to retain its military and technology advantage over potential adversaries (Footnote-Admiral Natter Article).

Today we are in the early stages of implementing and transitioning the Navy and Marine Corps to the Navy Marine Corps Intranet (NMCI), an outsourcing procurement that was awarded during October 2000. The objective for *"NMCI is an enterprise-wide network that will provide the Navy and Marine Corps with secure, universal and integrated access to voice, video and data information exchange services."* The NMCI goal parallels the IT-21 initiative to field an end-to-end Information Network Infrastructure for Naval war fighting and business functions.

On 28 August 2001, Vice Chief of Naval Operation (VCNO), issued a memorandum subject: *Software/Applications for [Navy Marine Corps Intranet] NMCI.* A key requirement identified in this memo was the need for "a simple but clear definition of what a web enabled Navy would be." As a result, numerous organizations came together to determine the high-level technical issues associated with web enabling the Navy. Soon after, between October and November 2001, key requirements and broad timelines were finalized. In December 2001, the VCNO chartered Task Force Whiskey (TFW) to perform a detailed analysis, and provide, a workable execution strategy for web enabling the Navy. On 31 January 2002, the final report was delivered providing a vision for operational, technical, and system architectures as well as a proposed implementation timeline. Task Force Web (TFWeb) was established to implement a web-enabled Navy, *"To provide integrated and transformational information exchange for both the ashore and afloat navy to take full advantage of Navy's IT21 and NMCI infrastructure investments."*

## 1.1 Executive Summary

The NMCI initiative was designed to provide guidance for software developers that outlines the operating environment in which legacy applications and systems must operate. Legacy application owners must modify their existing systems to comply with NMCI guidelines or be forced to operate outside the NMCI enclave. The NMCI Mission is quoted as follows: *“NMCI is an enterprise-wide network that will provide the Navy and Marine Corps with secure, universal and integrated access to voice, video and data information exchange services.”*

The architecture envisioned for NMCI consists of the architecture management services in the Figure 1-1, anchored by networks, platforms, system services and applications.

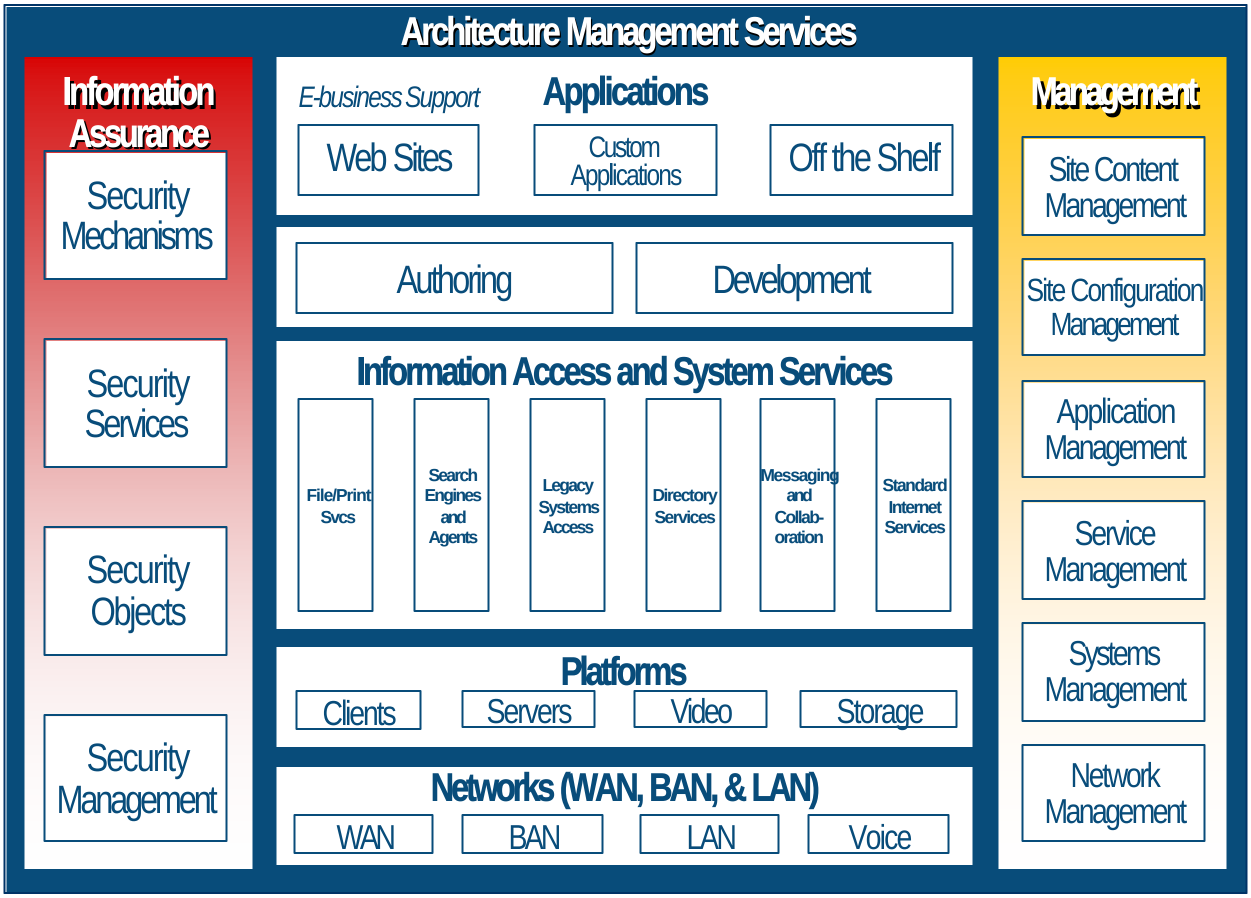


**FIGURE 1-1. ARCHITECTURE MANAGEMENT SERVICES**

Task Force Web (TFWeb) was established to implement the vision of a Web Enabled Navy. The mission of TFWeb is quoted as follows: “*To provide integrated and transformational information exchange for both the ashore and afloat navy to take full advantage of Navy's IT21 and NMCI infrastructure investments.*” This vision is being enabled by an enterprise three-tiered architecture describing where the different web technologies reside. The three tiers include presentation, application, and data (see Figure 1-2). The communication between the layers is based on standard Application Programmer Interfaces (API) and open-industry standards, such as XML. It also leverages various standard NMCI and TFWeb Afloat horizontal services. The proposed architecture is a comprehensive solution that provides a number of benefits, including

- Multiple physically distributed portals with only one logical portal

- Load balancing and clustering, providing scalability and high availability
- Open, flexible, distributed, and extensible architecture
- Minimizes risk against the use of new technologies
    - Strategic foundation for business and process integration, providing a consolidated view of the enterprise.

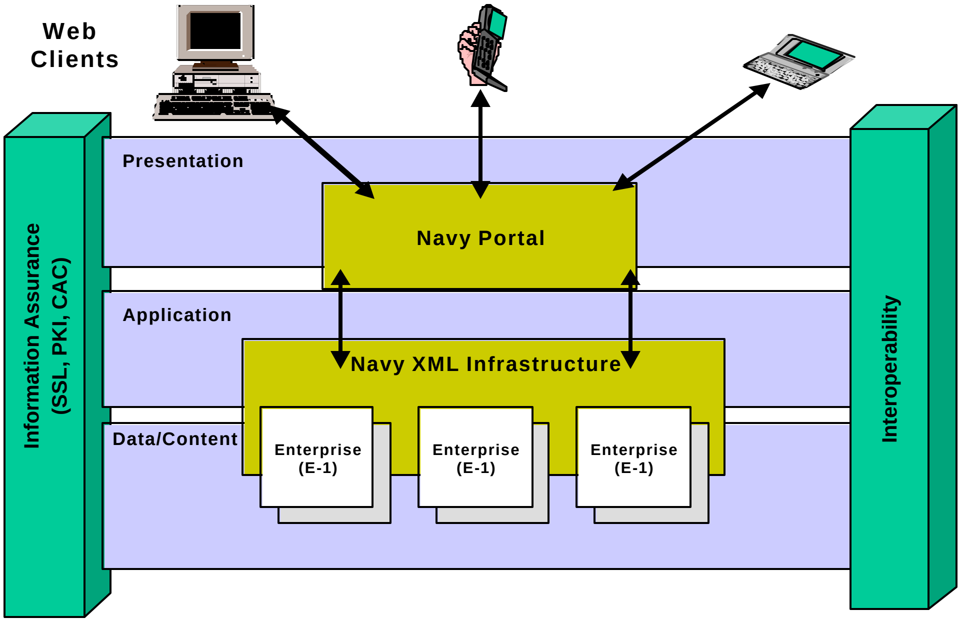


**FIGURE 1-2. THREE TIERS OF THE TFWEB PORTAL**

The primary capabilities of the web services architecture are

1. The ability to clearly establish the ***context*** of a users request for service. A users request context consists of a clearly established user identity and role, the level of authentication, and the ability to establish the delivery channel, or path, the request must traverse.
2. The ability to discover and execute services on behalf of a user or application request. This includes access to or hosting of the application or information services, as well as filtering based on the context of the request.
3. The ability to effectively manage the infrastructure, applications and information that are delivered through the web services framework.

Most client server applications are not accessed through the portal. A "fat" client is loaded on the desktop and is the primary means of accessing the application. In some cases, a client server application is hosted from a terminal server. The terminal services client can be accessed through the portal to establish a session with the terminal server.

As previously stated, NMCI has the goal of merging both client and web applications into one Network when the enterprise development is completed. Task Force Web's architecture also defines levels of integration for application developers. The Task Force Web initiative provides guidance for software developers that outlines how legacy applications must be integrated into the Navy Enterprise Portal. The difficulties encountered with both initiatives led to the analysis between the two sets of guidelines to determine if the two would be compatible or synergistic. TFWeb and PEOIT Enterprise Solutions formed a partnership based upon a common view of the end state architecture for the DON and a very real sense of urgency to document this view. The merger of the former Application Resource Guide and Task Force Web Developers Guide represents the understanding that the most coherent approach would be to provide application developers the tools that support Enterprise Application Management.

## 1.2 Objective

This guide communicates the technical and management direction an Application/Web Services Developer must follow to effectively develop or modify applications intended to operate on the NMCI. It is the intent of this guide to provide Navy service owners (i.e., developers, integrators, and implementers) of operational and business processes with detailed guidance while ensuring the seamless integration of existing service applications into the Enterprise Portal infrastructure. By providing sufficient information, this document seeks to reduce the time and cost of developing/modifying and fielding those applications. As a consolidated resource, the guide eliminates the confusion of multiple documents that address only portions of the application development process. Wherever this document does not provide specific direction, supplementary references are provided. It is assumed that all analysis and requirements gathering will be completed prior to initiating the development effort, therefore this guide will only focus on those technical elements required for integration of services into the portal. This document does not address the decision of when to create a service, or which service modules should be created to access specific application functionality.

It is assumed that the applications being considered for integration into the Enterprise Portal have been web enabled. This document focuses on the integration into the Enterprise Portal and does not describe detail level on how to web enable the application itself; however, it does provide developers with high-level items to consider prior to migrating their applications. As described in Section 2, there are increasing levels of integration, and simple web enabling of an application (i.e., access to an application through a web-browser client) only achieves hotlink integration with the Enterprise Portal, which is the least desirable level of integration.

## 1.3 Scope

This application developers guide addresses information and processes necessary to transition, modify and develop applications intended for use within DoN, operating on the NMCI. This guide provides a consolidated source of information, guidance and direction to developers who build or modify applications and/or the acquirers of applications intended for use within NMCI. Included are processes required for ISF certification, and security certification and accreditation (C&A), of applications prior to integration and operation on NMCI. Additionally, the document presents methods, processes, and interfaces for use by applications that extend beyond NMCI boundaries and other agency, service, contractor or joint applications that need to be accessed by NMCI users. This guide documents a rapidly evolving environment and is intended to be a work in progress with enhancements inserted as required to support the current state of NMCI implementation. As such, it is a work in progress and will change over time as the technology is enhanced and as additional integration issues are identified. Each section will be expanded as more information becomes available and guidance to developers /implementers is generated.

Information in the guide presently represents NMCI Release 1.0. and discusses the NMCI infrastructure architecture and the requirements necessary to develop and certify a new or emerging application for inclusion on the NMCI. It further, contains the checklists developers and/or acquirers of applications should follow as the application goes through its development or modification process. Additional sections address standards/programming practices, processes, and miscellaneous topics such as reusable components, metrics, and timelines. The guide concludes with resource appendices. Further, for the purposes of this Guide, Legacy Applications refer to any customer software application that exists prior to the Assumption of Responsibility (AOR) that is not included in the NMCI standard seat services or the CLIN 0023 catalog. The NMCI Legacy Application Transition Guide will be used for Legacy Applications, and provides a detailed look at the processes used to transition DON legacy applications into the NMCI environment.

This document will answer the question, “How do I develop my applications to be compliant with NMCI?,” “What can I do as a developer to assist with the ISF certification process?,” or “How do I get my application into the operational Enterprise Portal environment?” The following detailed guidance will be provided on performing the following activities:

- Registering a developer for access to the TFWeb environment.
- Registering and querying applications, data structures/fill, Object Interfaces /Application Programming Interfaces (APIs).
- Scheduling integration and operational testing.

The role of the TFWeb team with regard to the portal integration development process will be to

a) Establish registry requirements for applications served by the Enterprise Portal (e.g., applications, content).

b) Establish and track metrics to ensure

  - Timeliness and availability of updates for specific applications
  - Consolidation of applications

- Bandwidth and storage projection.

c) Generate and maintain the technology implementation/migration plan.

d) Establish security policy.

e) Act as Designated Approval Authority (DAA) for the Enterprise Portal testing, beta and pilot environments, as necessary.

### 1.3.1 Intended Audience

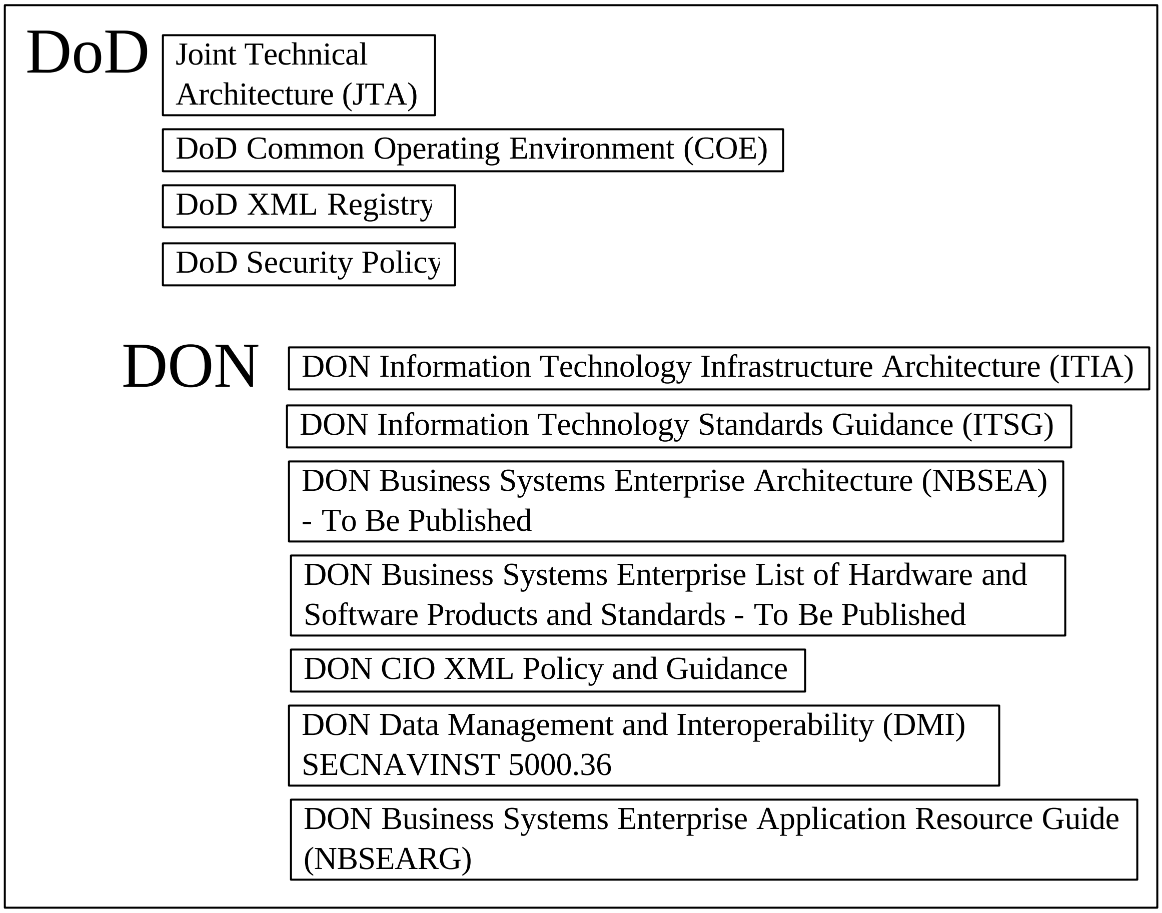
This guide is focused on the Presentation tier and its integration to the application/business logic tier. Other documents will address NMCI applications and provide the synchronized integrated guidance that addresses TFWeb/NMCI applications and web enablement. Procedures that lead to compliance with ITSG architectures and DoN standards will also be included. Thus, this guide is intended for the following audience:

- WEN Integration Architects/Engineers
- NMCI Integration Architects/Engineers
- WEN Application Developers
- NMCI Application Developers
- Subject Matter Experts (SMEs) involved in NMCI and Web enabled application development
- Application/Content Owners.

## 1.4 DoD and Navy Guidance Context

A challenge facing programs today is identifying and complying with policies, directives and mandates from multiple sources. Exacerbating this situation, programs find that some guidance, with which they must comply, appears to conflict. This [application resource guide] serves to provide one stop shopping to interpret the various sources of guidance and provide its context. Using this guide, programs can successfully deliver systems that meet their ORD requirements and also comply with DoD and DON CIO policies, directives and mandates.

Figure 1-3, Information Technology Guidance Context, provides a top level depiction of that guidance. Within the sections of this developers' document, guidance from the various sources depicted in this figure is presented. For example, the security section presents guidance from the DoD and Navy (i.e., Policy, COE, DITSCAP, Task Force Web and NMCI) sources in a logical manner to support complying with them all.



DoD

Joint Technical Architecture (JTA)

DoD Common Operating Environment (COE)

DoD XML Registry

DoD Security Policy

DON

DON Information Technology Infrastructure Architecture (ITIA)

DON Information Technology Standards Guidance (ITSG)

DON Business Systems Enterprise Architecture (NBSEA) - To Be Published

DON Business Systems Enterprise List of Hardware and Software Products and Standards - To Be Published

DON CIO XML Policy and Guidance

DON Data Management and Interoperability (DMI) SECNAVINST 5000.36

DON Business Systems Enterprise Application Resource Guide (NBSEARG)

**FIGURE 1-3. INFORMATION TECHNOLOGY GUIDANCE CONTEXT**

## 1.5 Assumptions

The following assumptions were made regarding the environment in which this Enterprise Portal will be implemented.

- **Scope –** The capabilities of the Enterprise Portal will be deployed to ashore and afloat organizations and facilities. The Enterprise Portal will leverage NMCI and TFWeb Afloat infrastructures. The Enterprise Portal provides access to the content and data, which are accessible through web-enabled applications.
- **Focus –** The focus of the Enterprise Portal is initially internal. That is, initial functionality will be developed and deployed to support NMCI business and warfare operations processes. The middle and long-term possibilities include using the portal system to inter-operate with Allies, coalition forces, commercial suppliers, retirees, and dependents. The Enterprise Portal will be structured to host or link to other service (Joint/non-DoD) applications whenever possible.

- **Implementation** – Implementation and deployment of web-service capabilities will be incremental; that is, delivery of functionality and content to the users will be evolutionary. This concept of operations (CONOPS) is expressed from the viewpoint of a steady state, fully functional Enterprise Portal system, where the information content of the system is ever changing.
- **Examples –** where possible, realistic examples are provided of source code. However, URLs used in these examples are not literal and should be replaced with correct references.
- **Commercial Standards-Centric –** Where possible, the Enterprise Portal will use commercial standards for its interfaces, underlying technologies, and applications. Some of the technologies and/or interfaces being considered for use include, but are not limited to, the following:
- Java/Java 2 Enterprise Edition (J2EE): Java was introduced in 1995 by Sun Microsystems. It is an object-oriented language designed for the World Wide Web (WWW), similar to C/C++, in which the source is compiled into 'bytecode', which is then interpreted by run-time environment (known as a Java Virtual Machine) on the host machine. J2EE is a java-centric environment for developing and deploying mult-tiered web-based applications. Some key features of J2EE include:
- Enterprise JavaBeans (EJB). EJB is a Java API developed by Sun that defines the component architecture for multi-tiered systems. EJBs are the objects in a multi-tiered object-oriented J2EE environment that enable the developers to focus on actual business architecture as opposed to developing the interfaces between the different components themselves.
- Servlets and Java Servlet Pages (JSPs). Servlets are java applets that run on the server side as an alternative to Common Gateway Interface (CGI) applications. JSPs are an extension of servlets that allow web developers to dynamically build web pages.
- eXtensible Markup Language (XML): XML is an extension/subset of Standard Graphical Markup Language (SGML) specifically designed for WWW dissemination and display of data. It is an open framework in which developers can develop (and, more importantly, standardize and validate against) a tagged data format. When done properly, the tagging becomes a form of metadata that can be used to more easily transition/translate the data inter/intra system or application. Furthermore, the rendering of XML is detached from the data itself. Therefore, the data in an XML document/file can be reformatted for display or processing any number of ways without ever having to modify the tags they contain.
- XML Schema: While XML 1.0 supplies a mechanism, the "Document Type Definition" (DTD) for declaring constraints on the use of markup, automated processing of XML documents requires more rigorous and comprehensive facilities in this area. The requirements are for constraints on how the component parts of an application fit together, the document structure, attributes, data-typing, and so on. XML Schema addresses the means for defining the structure, content and semantics of XML documents

- SOAP: SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined datatypes, and a convention for representing remote procedure calls and responses. SOAP can potentially be used in combination with a variety of other protocols; however, the only binding defined in this document is SOAP in combination with HTTP.
- UDDI: The Universal Description, Discovery and Integration (UDDI) specifications define a way to publish and discover information about Web services. The term "Web service" describes specific business functionality exposed by a company, usually through an Internet connection, for the purpose of providing a way for another company or software program to use the service.Even when one considers XML and SOAP, though, there are still vast gaps in implementing a communications infrastructure. The UDDI specifications borrow the lesson learned from XML and SOAP to define a next-layer-up that lets organizations share a way to query each other's capabilities and to describe their own capabilities.
- **NMCI** - The Navy Marine Corps Intranet (NMCI) will supply the shore-based and embarkable infrastructure.
  - **TFWeb Afloat –** TFWeb Afloat and related shipboard standards and delivery mechanisms will provide an afloat infrastructure.

# 2.0 Enterprise Architecture, ~~Infrastructure,~~ Web Services, Infrastructure, Standards/Practices

## 2.1 Enterprise Architecture

### 2.1.1 Architecture Background

The Department of the Navy (DON) Chief Information Officer (CIO) has established an IM/IT strategic plan, which outlines the mission, vision, and guiding principles designed to “ensure that our Sailors, Marines, Civilians, and Reservists have the right information, knowledge, and technology to successfully perform the DoN missions.” One of the key visions laid out in the document is the need to establish an effective, flexible, and sustainable DoN enterprise wide information and technology environment…” ~~Furthermore, this strategic~~The plan outlines as its first goal, the establishment of an IT infrastructure architecture. The DoN IM/IT Strategic plan can be located at www.don-imit.navy .

~~2.1.2 i One of the key visions laid out in the document is the need to establish an “effective, flexible, and sustainable DoN enterprise-wide information and technology environment…”ii Furthermore, this strategic plan outlines, as its first goal, the establishment of an IT infrastructure architecture. The DoN IM/IT Strategic Plan can be located at www.don-imit.navy.mil.~~

~~2.1.2 Baseline~~

### 2.1.2 Architecture ~~assumptions~~Assumptions

This document~~[ARG]~~ makes several ~~Several~~ architecture assumptions ~~have been made in this document. There are~~for several reasons ~~for stating these assumptions~~. In many cases, the document provides guidance in areas that have not yet been addressed at the policy level. It should also be noted that ~~Also,~~ Program Managers may have additional requirements beyond the scope of this first version, which must be taken into account. ~~In other areas, such as the role that the DII COE plays in a web services oriented architecture are being addressed but have not yet been resolved.~~ Those assumptions are:

- DoN CIO will develop and publish an~~n~~ ~~enterprise~~ Enterprise ~~architecture~~Architecture, including a set of Architecture Principles to be followed regardless of technology and standards evolution over time. ~~. Furthermore, t~~
- The DoN ~~CIO a~~Enterprise Architecture will be consistent with this document.
- Program managers~~/software developers~~ will ~~be~~ developing ~~software components~~systems that ~~will be hosted on platforms that~~ meet their Operational Requirements Document (ORD), including existing higher echelon architecture requirements such as compliance with the JTA and ~~DII~~DoD Common Operating Environment.
- [THIS MIGHT NEED TO BE MOVED TO "SCOPE" IN SECTION 1] This document applies ~~only~~ to all new programs and to those legacy applications that have been approved for continuation via the Navy’s ongoing application reduction initiatives.

### 2.1.3 Architecture Purpose.

While work is beginning on the development of a ~~Navy~~ DoN Enterprise Architecture (DoNEA) ~~architecture~~, this document attempts to outline the high level architectural requirements deemed essential to integrate with Task Force Web and NMCI that will support the Navy’s movement towards a ~~Web~~ web ~~Services~~ services-o~~Oriented~~ riented ~~Architecture~~ architecture~~(WSOA)~~. In a future iteration this section will be expanded to document architectural implications of additional ORD-specified requirements facing programs, including the DoD JTA and ~~Common Operating Environment (~~COE~~)~~.

### 2.1.4 Architecture Scope

~~This a~~G~~rchitectural direction~~ uidance provided in section 2.0 ~~focuses on the providing~~ provides technology selection advice and discusses web services, infrastructure and standards. ~~using open, standards-based protocols and interfaces.~~ ~~While some protocols for the development of web services oriented architectures have yet to be completed, it is the Navy’s intention to adopt these standards as soon as they are commercially acceptable and are capable of meeting the Navy’’s needs.~~ This architectural guidance ~~serves as the foundation for~~supports Program Managers / Central Design Authorities (CDAs) to provide component-based information systems ~~to~~ that deliver required services across distributed, heterogeneous platforms, hiding the complex distribution issues from both the end-user and the applications developer. The architecture exploits the benefits of encapsulation and component reuse~~,~~ and ~~it~~ promotes designs that interoperate with legacy and other external systems. It also facilitates system maintenance, enhancement, management, and a high degree of security and reliability.

### 2.1.5 Architecture Technology Selection

Critical decisions face Program Managers and other Central Design Authorities (CDAs), tasked to design/develop/deploy new systems or evolve existing legacy systems. Among them are architectural decisions to select technologies that will facilitate certification under Task Force Web, NMCI and DoD JTA and COE requirements. These decisions need to take into account life cycle cost implications, such as those brought about by technology evolution~~ving~~ and the level of vendor~~s~~ ~~emerging or stopping~~ support for legacy and emerging~~of~~ products. For example, if a Program/CDA had to replace one system component for any reason, then they should not be forced to replace others or the whole architecture.

Guidance in this respect should be addressed in a DoN Enterprise Architecture, addressed in assumptions above. Also, Enterprise Domain Architects may have decided upon technologies for use with all systems within their domain. An example would be the selection of the J2EE Distributed Object Computing (DOC) technology for use within the Navy Personnel Domain. However, in the absence of guidance from a DoN Enterprise Architecture or an Enterprise Domain Architect, Program Managers/ CDAs are encouraged to perform trade studies that consider all requirements to make prudent decisions. This document~~[ARG]~~ provides references to some requirements sources in the last paragraph of this section. Also, a wealth of information to support technology trades is available by Gartner and other online sources.

One of the more crucial decisions regards which DOC technology to use. J2EE and .Net are current options. Both options are consistent with Task Force Web, NMCI and DoD JTA and COE requirements and should be included in trade studies.

### 2.1.6 Security Architecture

For address of security aspects of architecture, refer to the Information Assurance Section 6.0.

## 2.2~~2.1.5~~ Web Services

## 2.2.1 Definition

Several definitions of web services are provided below.

### 2.2.1.1

According to Graham Glass, author of The Web Services (~~r~~r~~R~~)~~E~~Evolution, a web service is a “collection of functions that are packaged as a single entity and published to the network for use by other programs. Web services are building blocks for creating open distributed systems, and allow companies and individuals to quickly and cheaply make their digital assets available worldwide.”

### 2.2.1.2

Gartner Group describes web services as “loosely coupled software components that interact with one another dynamically via standard internet technologies.”

Web services provide a lightweight interface to and from systems using open, standards-based protocols~~that comply with this architectural guidance~~. They make it possible to expose all business functions provided by the architecture with a low barrier of technology compatibility. Through the use of XML and the HTTP and SOAP protocols, web services can be provided irrespective of the language and technology of both the accessing system and other systems within the ~~Domain~~ architecture. Web services effectively hide the internal organization of the ~~Domain~~ architecture and implements distributed object computing. There is no need to use mechanisms such as CORBA, DCOM or RMI to access remote methods in the ~~Domain~~ architecture. Any part of the ~~Domain~~ architecture API can be exposed in the form of a web service. It is possible to expose a method as a web service, through RMI or through CORBA in response to a variety of functional and performance requirements.

### 2.2.2 Benefits

~~A Web service[iii] is a collection of functions that are packaged as a single entity and published to the network for use by other programs. Web services are building blocks for creating open distributed systems. A Web service can aggregate other Web services to provide a higher-level set of features.~~

~~Web services provide the following benefits to the Domain architecture:~~

~~Interoperability. Any Web service can interact with any other Web service. Thanks to SOAP, the new standard protocol supported by all of the major vendors (and most of the minor ones too), there is no need for converting between protocols such as CORBA or DCOM. Web services can be written in any language.~~

- Interoperability. Any Web service can interact with any other Web service. Thanks to SOAP, the new standard protocol supported by all of the major vendors (and most of the minor ones too), there is no need for converting between protocols such as CORBA or DCOM. Web services can be written in any language
- Ubiquity. Web services communicate using HTTP and XML. Therefore, any device that supports these technologies can both host and access Web services.
- Low barrier to Entry. The concepts behind Web services are easy to understand and free toolkits from vendors like IBM and Microsoft allow developers to quickly create and deploy Web services. In addition, some of these toolkits allow pre-existing COM components and JavaBeans to be easily exposed as Web services.

Industry Support. All of the major vendors are supporting the current and emerging web services protocols such as XML and SOAP.

~~Industry Support. All of the major vendors are supporting SOAP and the surrounding Web services technology.~~

- Encourages reuse of previously developed components, therefore reducing total development cost to the enterprise.

Reduced complexity by using encapsulation to hide the implementation details of the services~~[This section moves to the interfaces section]~~ ~~2.1.6 Simple Object Access Protocol~~

~~In the W3C Simple Object Access Protocol (SOAP) 1.1 Note of May 8, 2000, SOAP is defined as follows. SOAP provides a simple and lightweight mechanism for exchanging structured and typed information between peers in~~

~~a decentralized, distributed environment using XML. SOAP does not itself define any application semantics such as a programming model or implementation specific semantics; rather it defines a simple mechanism for expressing application semantics by providing a modular packaging model and encoding mechanisms for encoding data within modules. This allows SOAP to be used in a large variety of systems ranging from messaging systems to RPC.~~

~~SOAP consists of three parts:~~

~~? The SOAP envelope construct defines an overall framework for expressing what is in a message, who should deal with it, and whether it is optional or mandatory.~~

~~? The SOAP encoding rules defines a serialization mechanism that can be used to exchange instances of application-defined datatypes.~~

~~? The SOAP RPC representation defines a convention that can be used to represent remote procedure calls and responses.~~

~~Although these parts are described together as part of SOAP, they are functionally orthogonal. In particular, the envelope and the encoding rules are defined in different namespaces in order to promote simplicity through modularity.~~

**~~[Need to replace the sections 2.1.7 and 2.1.8 with one that:~~**

**~~? Addresses Thursday teleconference conclusion: "We discussed that DOC technology really needs to be addressed in a Navy Enterprise Architecture artifact and that the DOC technology selected may need to be the decision of each Domain Architect within the Navy, based on consideration of Architecture Principles."~~**

**~~? Identify the two Distributed Object Computing alternatives and the Architectural Principle implications of the Domain decision to select each one.~~**

**~~? Provides references to source materials.~~**

**~~? Provides advice that if they embrace these alternatives that they have a reasonable expectation of integrating with NMCI and TFW]~~**

### ~~2.1.7 Enterprise JavaBeans Component Model~~

~~What is a web service?~~

~~A Web service is a "collection of functions that are packaged as a single entity and published to the network for use by other programs. Web services are building blocks for creating open distributed systems, and allow companies and individuals to quickly and cheaply make their digital assets available worldwide."[iv] Gartner Group describes web services as "loosely coupled software components that interact with one another dynamically via standard internet technologies."~~

~~2.1.5 What are the benefits of adopting a web services architecture?~~

~~? Greatly simplifies integration of legacy applications by using wrapper technology.~~

~~? Improved interoperability due to the use of components, which communicate via standard protocols such as SOAP and XML.~~

~~? Reduced complexity by using encapsulation to hide the implementation details of the services.~~

~~? Widespread industry support for the open standards being adopted to support web services.~~

~~? New functionality can be developed and deployed quicker than existing architectures.~~

- ~~Encourages reuse of previously developed components, therefore reducing total development cost to the enterprise.~~

~~2.1.6~~ There are many reasons for adopting a web services oriented architecture. The most significant reason is to support an orderly migration of the Navy's vast number of legacy applications into the NMCI and TFW architectures. Using industry standard protocols such as XML, SOAP, and UDDI, it offers a cost effective and timely mechanism for exposing business functionality contained within existing monolithic systems via web interfaces.

The diagram below (figure 2-1), provided by The Stencil Group, identifies the protocol stack, indicating which web services standards are currently in force or emerging. The Navy intends to adopt the emerging standards as they become commercially acceptable and are capable of meeting the Navy's needs.

| Layer | Type |
|---|---|
| Other Business Rules (undefined) | Emerging Layers |
| Web Services Flow Language (WSFL) | |
| Universal Description, Discovery and Integration (UDDI) | |
| Web Services Description Language (WSDL) | |
| Simple Object Access Protocol (SOAP) | Core Layers |
| Extensible Markup Language (XML) | |
| Common Internet Protocols (TCP/IP, HTTP) | |

Figure 2-1 Web Services Protocol Stack

Source: The Stencil Group

## 2.3~~2~~ Infrastructure ~~Web Services Infrastructure~~

~~2.32.1 The Navy's Web Services Infrastructure will not specify implementation details, such as selecting a specific framework such as J2EE or Microsoft's .NET, but will specify interface standards and protocols that will ensure the ability of components to communicate with each other to maximize interoperability. This guidance does not preclude the specification of such frameworks by lower level organizations that seek to maintain consistency within their functional or domain area.~~

~~An example would be the adoption of the J2EE framework for use within the personnel domain as managed by the SPAWAR Information Technology Center (SITC). The diagram below (figure 2-1), provided by The Stencil Group, identifies the protocol stack for a web services architecture and indicates which standards are currently in force, or emerging. It shall be the Navy's intent to reach the objective architecture by adopting the emerging standards, as they become commercially acceptable.~~

| Layer | Type |
|---|---|
| Other Business Rules (undefined) | Emerging Layers |
| Web Services Flow Language (WSFL) | |
| Universal Description, Discovery and Integration (UDDI) | |
| Web Services Description Language (WSDL) | |
| Simple Object Access Protocol (SOAP) | Core Layers |
| Extensible Markup Language (XML) | |
| Common Internet Protocols (TCP/IP, HTTP) | |

~~Figure 2-1 Web Services Protocol Stack~~

~~Source: The Stencil Group~~

### 2.3~~2~~.~~2~~1 Task Force Web and Web Services

Task Force Web is putting the web services infrastructure in place for the Navy. The Task Force has firmly established the use of XML and SOAP as an interface between components and has moved towards establishing a UDDI registry for the Navy. Task Force Web's Service Registry will migrate to the UDDI standard in the near future.

### 2.3~~2~~.~~3~~2 NMCI and Web Services

By using the web services infrastructure, it should be easier to migrate legacy applications into NMCI. This should be possible for several reasons. Since the transport interface is defined as SOAP/XML, which is firewall-friendly, services provided by the legacy application should be more widely available to NMCI users, and also easier to use. Developers should find it easier to combine available services to create new functionality. Deployment of new services should occur more quickly because standardized application servers will be provided by NMCI. Because the developer is only providing software components, the NMCI certification process can be streamlined. In the short term, application owners that are concerned about large portions of their customers losing access to the application due to the NMCI rollout, it may be possible to rectify this situation by developing SOAP interfaces directly between the TFW portal and the existing application. This would allow the customer base to access the application (or portions of it) even though the application resides within the NMCI enclave and the customers have not yet migrated to NMCI.

### 2.3~~2~~.43 IT-21 and Web Services

As part of the TFW initiative, efforts are underway to web enable and provide access via the TFW portal to all existing applications that are used aboard ship. As such, TFW is putting hardware and software aboard ship that will support those application developers with migrating their functionality into web services. The TFW architecture includes an application server (currently BEA WebLogic) that could be leveraged to host the components (.NET or J2EE) supporting these services. This would serve to reduce the hardware costs of the developer and reduce the footprint requirements on the ship.

## 2.4~~3~~ Process for developing web services

This section will discuss two approaches for developing web services. The first discussion will provide ~~high level~~high-level guidance for developing new functionality or services. The second discussion will discuss how to create web services from existing legacy applications. More detailed information on this subject can be found in the Migration Planning section of the document.

~~2.3.1 Baseline assumptions. Several assumptions have been made in this document. There are several reasons for stating these assumptions. In many cases, the document provides guidance in areas that have not yet been addressed at the policy level. In other areas, such as the role that the DII COE plays in a web services oriented architecture are being addressed but have not yet been resolved. Those assumptions are:~~

- ~~? DoN CIO will develop and publish an enterprise architecture. Furthermore, the DoN CIO architecture will be consistent with this document.~~
- ~~? Program managers/software developers will be developing software components that will be hosted on platforms that meet existing higher echelon architecture requirements such as JTA and DII COE.~~
- ~~This document applies only to those legacy applications that have been approved for continuation via the Navy's ongoing application reduction initiatives.~~

### 2.4~~3~~.1 New Services

- Identify a service that is required by existing or future users.
- Search the enterprise registry to determine if a similar service already exists and is suitable.

- Define the SOAP/XML request and response messages for the service. (This definition requires describing all arguments required by the service, and well as all data returned by the service.)
- Implement a module that accepts the SOAP request message, performs the indicated service, and generates the SOAP response message.
- Submit the module to the appropriate registration authority for publishing in the enterprise service registry. (Registration authority will conduct testing and verification prior to publication).

### 2.4~~3~~.2 Web Services from legacy applications

- Identify key functionality that is required to support existing and future users of the application.
- Decompose the application into discrete services that provide the functionality identified above.
- For each service, search the enterprise registry to determine if similar service already exists and is suitable.
- For each service, define the SOAP/XML request/response messages.
- Implement a module that accepts the SOAP request message, performs the indicated service and generates the SOAP response message.
- Test the service
- Submit the module to the appropriate registration authority for publishing in the enterprise service registry. (Registration authority will conduct testing and verification prior to publication).

## 2.5 References

~~This section has briefly discussed several technologies and protocols.~~ Additional information on these topics addressed in this section can be found at the following web sites:

- J2EE http://java.sun.com/j2ee/
- .NET http://microsoft.com/net/
- Web Services
  - http://webservices.org/
  - http://www.xml.com/pub/a/2001/04/04/webservices
- UDDI http://www.uddi.org/
- XML http://www.xml.org/
- SOAP http://www.w3c.org/tr/soap/
- COE http://diicoe.disa.mil/coe/
- JTA http://www-jta.itsi.disa.mil/

## 2.6 Standards and Practices

The NAVY is drastically changing toward the use of today's most innovative IT technological standards. The NAVY Web Enablement Initiative is designed to coordinate, utilize, leverage and bolster resources, including current and projected expenditures, in order to meet the CNO mandated plan to web-enable the NAVY, known as the Task Force Web (TFWeb). These requirements are in the areas of web services, XML and other emergent technologies associated with web enablement. Success of the Navy being web enabled is dependent on an efficient, uniform and consistent process to transform NAVY from an application-centric to an information/service-centric Enterprise, via the web services architecture. To accomplish this all Navy Applications must be HTML-enabled with data being rendered through the use of XML. Style sheets will be used to express the data in the business logic needed by the user of the service that is being provided. All applications or data must be decompressed into Web Services for integration into the Navy Enterprise Portal.

The Navy Architecture and Infrastructure depends on standard configurations of networks, servers, data, services and workstations to achieve the goals the DoN has set for it. Therefore, applications developed for, existing on, or accessed through NMCI assets must comply with the architectures, standards, protocols, and constraints (such as NMCI naming conventions, TFWeb Levels of Web Enablement) imposed by these networks, services, servers and workstations. (Sections 3and 6). In addition, the DoD and the DoN have established additional standards, protocols, and architectures that are recommended for use as needed when modifying, developing, or evolving DoN applications (JTA, ITSG, GIG, etc.). The Navy strongly recommends that modern programming practices and procedures be used during the modification, development, or evolution of applications intended for operation on, or interaction with, NMCI. Examples of these practices can be found in ANSI, IEEE, and ISO Software Practices and Principles. They can be found at the following URLs:

1. https://www.infosec.navy.mil
2. http://www.DISA.dod.mil/dii_coe
3. http://www.eds.com/nmci/legacy_applications_transition_guide.doc

The following sections describe standards that must be adhered to for Web Enablement and the integration of all applications into the Navy Enterprise Portal.

### 2.6.1 Web Services

The process for devolving an application into Web Services requires some reverse engineering of the application. Applications are usually developed from a Product Requirements Document (PRD) that details what the application is supposed to accomplish for the user and owner. From this document the functional elements (tasks) are determined. These functional elements are or will become the Web Services that the application will supply. Web Services are the business logic/rules that an application uses, not the presentation. A Level of integration for this Web Service can be selected from this process and presented by a Service Module through the Navy Enterprise Portal. (Section 1.0)

### 2.6.1.1 Service Module Definition

A Service Module is a lightweight application connector that reveals some piece of application functionality and makes it available, in a web-enabled format, to end-users through the portal. A Service Module does not contain application business logic. Application business logic and data continue to reside within the application or its existing data store.

Application owners are responsible for creating and maintaining the Service Modules and their back-end applications and services. These may be created using any capable application server platform or language, but they must conform to the TFWeb requirements.

### 2.6.1.2 Levels of Integration

Levels of integration are defined to be the degree of integration between a service or application and the enterprise portal. Web enabling of an application will provide some degree of access to an application through a web browser, but the levels of integration defined here describe the degree to which the web-enabled application has been integrated into the portal. Application/Data Integration is the desired level of integration; however, there may be justifiable reasons why that level of integration cannot be achieved or does not make sense for a particular application. Rationale will need to be provided for applications that will only achieve Hyperlink or Presentation levels of integration.

Each level of integration defined below will have specific requirements for integrating with the enterprise portal. While this section defines the levels of integration and the integration goal, Section 3.2.1 will describe the integration requirements and process for each of these levels of integration. Please note that this discussion refers to the integration of services/applications with the TFWeb Portal.

- **Level 1 – Hyperlink Integration**
  Hyperlink Integration provides "as is" access to an existing web-based application through a hyperlink or a list of hyperlinks displayed within a pane of the portal. The hyperlink is created as a service within the service repository, with access to this service controlled through the permissions management application. When accessed, the hyperlink is displayed within a pane of the portal on the user's desktop, and generally includes a brief description of the application being accessed, and possibly an associated lightweight graphic. When the user selects the hyperlink, a new browser window is opened on the user's desktop, through which the application will execute. At this point, communication occurs directly between the new browser window and the application.

  This method is usually easiest to implement, though it is the least desirable. Application owners attempting this level of integration must first obtain a waiver from TFWeb, and must also provide a migration plan for achieving a higher level of integration. Hyperlink Integration requires no true integration with enterprise portal services or content, other than the initial authentication that the user has access to the service containing the hyperlink. Therefore, there is minimal benefit gained in its integration into the enterprise portal. When working with multiple applications using hyperlink integration, multiple browsers will be opened on the user's desktop.

  Hyperlink Integration is commonly referred to as Level 1 Integration.

- **Level 2 – Presentation Integration**
  Presentation Integration provides "as is" access to an already web-enabled application. This level of integration requires that all application content be rendered within a pane of

the portal. The initial connection to the application is created as a service module in the Enterprise Service Repository utilizing a standard HTTP redirect to the application. Access to the service module is controlled through the permissions management application. When accessed by the user, the application is displayed within a pane of the portal on the user's desktop. The user is able to directly interact with the application appearing in this pane. All communication between the user and the application must flow through the portal. This type of communication is implemented through a reverse HTTP proxy mechanism within the enterprise portal, and requires that the application present its content in portal compliant HTML or XML/XSL. The definition of portal-compliant HTML or XML/XSL is defined in a later section of this document.

Application owners attempting this level of integration do not need to obtain a waiver from TFWeb, but they must provide a migration plan for achieving a higher level of integration.

Presentation Integration allows multiple applications to be visible within multiple panes (e.g., channels, frames) of the same browser window of the enterprise portal. However, the content and data access mechanisms still reside outside of the service repository, and therefore may not match the presentation rendering guidelines of TFWeb.

Presentation Integration is commonly referred to as Level 2 Integration.

- **Level 3 – Application/Data Integration**

Application/Data Integration involves a more closely coupled integration of the application with the enterprise portal. Application/Data Integration is the TFWeb-preferred level of integration. All application content is provided through services that reside in the service repository. These types of services act as lightweight connectors, exposing some portion of application functionality in a manner that is compliant with the enterprise portal. Application logic continues to reside within the application/data layers, and not within the service. When accessed by a user, all application content is rendered within a pane of the portal on the user's desktop. Access to all services is controlled by the permissions management application. The user is able to directly interact with the application appearing in this pane. All communication between the user and the application must flow through both the portal and the service repository.

When invoked, a service decomposes the request, accesses the appropriate application or data source to process the request, formats the results of the request into the appropriate portal-compliant HTML or XML/XSL response, and returns the response to the enterprise portal. The definition of portal-compliant HTML or XML/XSL is provided in a later section of this document. A description of the service will be registered with the global service registry to provide the enterprise portal with quick access and search capability. Section 8 provides service developers with additional details of how to build Application/Data Integration services.

Application/Data Integration is commonly referred to as Level 3 Integration.

- **Capability Comparison Across Integration Levels**

There are a number of capabilities that are desirable for the Enterprise Portal, and the level of integration achieved for an application or service may impact the degree to which a particular capability can be achieved. A finite set of desirable capabilities are defined as follows:

1. Single Sign-on – The ability for users to authenticate once to the portal and be able to access all authorized resources within the enterprise. A single point sign-on accepts the user's name and password and automatically logs on to all appropriate services.

2. Access to Service – The ability to obtain access to a service from the portal.

3. Service Presented in Portal – The ability to view/manipulate some portion of the service's information from within the portal's presentation with the assumption that multiple services can be viewed in the same portal presentation at the same time.
4. Policy-enforced Look-and-Feel – The manual enforcement of standards by publishing policies and manually checking for compliance.
5. Automatically enforced Look-and-Feel – The enforcement of standards by being the single point of control. In the case of the portal, if the portal is uniquely responsible for all presentation of information, then ensuring that the portal's presentation meets the standard can automatically enforce the look-and-feel.
6. Data Sharing (cut-n-paste) – Data or information from one service is simply captured and then pasted into another service with the user taking responsibility for the format and definition of the data (e.g., cutting and pasting a text string). There is no understanding of the data by the portal. The portal processes information presentation (e.g., HTML) without having to understand the underlying data definition.
7. Data Sharing (data aggregation) – The portal processes the information and is responsible for formatting the presentation (e.g., XML style sheets) and provides the potential for additional business logic that manipulates dat a from multiple sources and data aggregation.
8. Business Process Integration – The re-engineering of existing business processes through the aggregation of services and data.

Table 2-4 shows the degree to which these capabilities can be achieved across the levels of integration. Clearly, application/data integration is the goal, but significant capability can be achieved with presentation integration. While hyperlink integration is not desirable, it may be an acceptable first step from web enabling to portal integration.

**Table 2-4. Capabilities By Levels of Integration**

| Capabilities | Hyperlink | Presentation | Application |
|---|---|---|---|
| Single Point Sign-on | ✓ | ✓ | ✓ |
| Access to Service | ✓ | ✓ | ✓ |
| Service Presented in Portal | | ✓ | ✓ |
| Policy-enforced Look-and-Feel | ✓ | ✓ | ✓ |
| Automatically-enforced Look-and-Feel | | **Limited** | ✓ |
| Data Sharing (cut-and-paste) | **Limited** | ✓ | ✓ |
| Data Sharing (data aggregation) | | | ✓ |
| Business Process Integration | | | ✓ |

- **Integration Level Requirements Summary**

  Some of the most significant requirements for each level of integration are summarized in Table 2-2. These requirements will be discussed in further detail throughout this document, see the document sections that are referenced in Table 2-5 for detailed information on each requirement.

## Table 2-5.  Integration Level Requirements

| Requirement<br>(Section Reference) | Integration Level | | |
|---|---|---|---|
| | **1. Hyperlink** | **2. Presentation** | **3. Application/Data Integration** |
| Service Module Implementation<br>Section 2 and 7 | Static HTML or XML/XSL file with lightweight images and one or more hyperlinks | ASP, JSP, or CGI containing HTTP Redirect | ASP, JSP, or CGI with PRI and SOAP interface handlers |
| PRI Interface (HTTP Header Message) Support<br>Section 7, Appendix B, (Code Samples) | No | No | As required by the application for session management and error reporting |
| SOAP Interface Support<br>Section 7 | No | No | Required – SOAP Client in service module, SOAP Server in the application |
| Service Module Security<br>Section 6 | Enterprise SSO Integrated, URI protected by application ACL | | |
| Application Security<br>Section 6 | Prompt for username and password unless the application is integrated with the Enterprise SSO. | | |
| Preferred Application Security Challenge<br>Section 6 | None | Username and Password in HTML Form in the application | Username and Password in HTML Form in service module. HTTP BASIC authentication over SSL, or better, is required in the application |
| EMS to Application Communications<br>Section 6 | HTTPS (128 bit SSL, DoD PKI Server Cert.) | | |
| Support for 1 way or 2 way SSL in Application Communications<br>Section 6 | Per Application Requirements | | |
| Portal Connector Reverse Proxy (URL Rewrite)<br>Section 3 | No | Required | Required |
| HTML BASE TAG for relative references<br>Section 3 | No | No | Required only for application backend references |

| **Requirement**<br>**(Section Reference)** | **Integration Level** | | |
|---|---|---|---|
| | **1. Hyperlink** | **2. Presentation** | **3. Application/Data Integration** |
| Portal Rendering of XML/XSL to HTML (XSLT)<br>Section 7 | Supported in Portal Connector for Service Module XML/XSL only (Xalan-Java version 2.2.D11) | Supported in Portal Connector (Xalan-Java version 2.2.D11) | Supported in Portal Connector (Xalan-Java version 2.2.D11) |
| EMS XML Parsing Support<br>Section 7 | Not Applicable | Not Applicable | Support is provided for: ASP: MSXML 4.0<br>BEA: Xerces Java 1.4.4 XML |
| EMS SOAP Support<br>Section 7 | Not Applicable | Not Applicable | Support is provided for: ASP: MS SOAP Toolkit Version 2.0 SP2<br>BEA: WebLogic Web Services |
| Portal IFRAME compatibility<br>Section 3 | No | Required | Required |
| EMS Data Storage and Replication or Update<br>Section 4.2 | Not Available | Not Available | Not Available |
| EMS Hosted Applications<br>Section 4.2 | Not Available | Not Available | Not Available |
| Service Module Output<br>Section 7 | HTML or XML/XSL | HTML or XML/XSL | XML/XSL |
| Application Output<br>Section 7 | HTML or XML/XSL | HTML or XML/XSL | One or more SOAP/XML service interfaces |
| Mobile Code (Applets, ActiveX, client-side script)<br>Section 6 | Allowed within TFWeb policies and guidelines | | |
| Client-Side Script<br>Section 7 | Supported | Supported with the following restrictions:<br>• Frame refs can not refer to '_top' frame<br>• No dynamically generated URL links | Supported with the following restrictions:<br>• Frame refs can not refer to '_top' frame<br>• No dynamically generated URL links |

| Requirement<br>(Section Reference) | Integration Level | | |
|---|---|---|---|
| | 1. Hyperlink | 2. Presentation | 3. Application/Data Integration |
| Application Frames/Iframes<br>Section 7 | Supported | Supported with the following restrictions:<br>• Cannot target '_top' frame. Frame refs should be named | Supported with the following restrictions:<br>• Cannot target '_top' frame. Frame refs should be named |
| Popup child windows<br>Section 7 | Supported | Not recommended, must identify application in title bar | Not recommended, must identify application in title bar |
| Maintain State in EMS module<br>Section 7 | NA | Not supported, session ID is provided | Not supported, session ID is provided |
| Cascading Style Sheets<br>Section 3 | Supported | See style-sheet reference | See style-sheet reference |
| Waiver Required<br>Section 11 | Required | No | No |
| Level 3 Migration Plan Required<br>Section 11 | Required | Required | No |

### 2.6.2 Standards and Practices for the Navy Enterprise Architecture

The Navy has accepted the following industry standards and practice for the implementation and integration towards a web-enablement and portal integration of a Web Service oriented architecture.

## 2.6.2.1 XML

The Department of the Navy will fully exploit Extensible Markup Language as an enabling technology to achieve interoperability in support of maritime information superiority. XML is a proven technology focused on a common method for describing information exchange parcels to achieve true interoperability is rapidly sweeping the Internet world. This technology, the Extensible Markup Language (XML), is radically transforming approaches to capturing, storing, processing, and exchanging information. XML is a meta-language to define other languages. In particular, it can be used to define languages that serve as a means of exchanging data between application systems across the Internet.

At its inception, XML provided a method for identifying and exchanging data. However, XML has become much more. Among other things, the components of the XML "family of standards" provide a framework for creating data models (XML), formatting XML data for output to different devices (XSL--print, audio, web, cell phones), parsing the models to extract data for processing (DOM), and linking XML-components (XLink, XPointer, XPath). The biggest activity around XML is the development of business standards, and XML enabled end-user services and applications. The two together now comprise a framework for achieving the global Internet based network of tomorrow. XML is being used with databases, with Web pages, with Web services, and as the basis for exchange protocols. XML enables new paradigms for achieving transportability of data.

XML's ability to provide a platform and application neutral format for preserving archived information against time is being leveraged to ensure data archived in an XML format will be readily available years later to new and different applications and databases. Because XML is platform and operating system independent, its application is enabling transportability of information management across systems. Additional new and exciting uses of XML in creating the truly interoperable ubiquitous global computer network are being discovered and implemented almost daily. The true power of this technology is in its ability to create application-to-application and application-to-human interoperability through a standard suite of protocols. In addition, the application of XML may reduce or eliminate the need for costly middleware systems and services to achieve data transformation.

In short, XML's growing set of information exchange protocols and ever-expanding suite of use concepts are ideally suited–and, in many cases, specifically developed, to support each functional element of application-to-application and application-to human-interoperability. XML is rapidly becoming an integral part of the vast majority of software applications already being introduced into the Navy architecture, from simple office automation to complex database and knowledge management applications. The broad support for XML—by developers, vendors, and users alike—as a robust and platform-independent data-description language has resulted in broad support for, and acceptance and adoption of, XML-based specifications and protocols as formal standards.

### 2.6.2.1.1 XML Technical Specifications

The World Wide Web Consortium (W3C) suite of XML technical specifications focuses on key areas within the global network computing architecture. Each of these facets will play key roles in the Navy enterprise architecture as well. The following summarizes the different facets of the XML technical specifications and their role in the developing Navy enterprise architecture

**Data Definition:** The core XML standard defines the rules for creating and using data markup (descriptors) to define data.
**Content Management:** The XML Linking Language (XLink), XML Pointer Language (XPointer), XML Path Language (XPath), XML Query Language (XQuery) and XML Namespaces provide methods for linking, referencing, extracting, transforming, and associating XML defined data.
**Output Management:** The Extensible Stylesheet Language (XSL) defines how XML defined information will be presented in web pages, paper media, or transformed between formats. XSLT relevant for transform to HTML.
**Structure Management:** The Resource Description Framework (RDF) provides an XML based lightweight ontology system for describing entity relationships and exchanging machine-understandable information. XML Schema defines "the structure, content and semantics of XML documents (W3C)."

**Processing Management:** The Document Object Model (DOM) provides a web based application-programming interface (API) for processing object hierarchies.
**Protocols:** The W3C XML Protocol activity and its Simple Object Access Protocol (SOAP) precursor define how multiple peers will exchange encapsulated XML information packets for remote procedure calls and document exchanges in a distributed environment.
**Security:** XML Encryption, XML Digital Signature, and XML Key Management address crucial pieces of security. These key security specifications, in conjunction with the electronic Business XML (ebXML) secure and reliable messaging transport specification, make XML particularly attractive in meeting today's security challenges.

### 2.6.2.2 SOAP 1.1

In the W3C Simple Object Access Protocol (SOAP) 1.1 Note of May 8, 2000, SOAP is defined as follows. SOAP provides a simple and lightweight mechanism for exchanging structured and typed information between peers in a decentralized, distributed environment using XML. SOAP defines a vocabulary in XML that allows heterogeneous components to collaborate to perform services. The use of XML as the data format for the interfaces to SOAP messages means that implementers are free to represent the data as they see fit in the language of their choice. Generic XML tools can also be used to support SOAP. SOAP does not itself define any application semantics such as a programming model or implementation specific semantics; rather it defines a simple mechanism for expressing application semantics by providing a modular packaging model and encoding mechanisms for encoding data within modules. This allows SOAP to be used in a large variety of systems ranging from messaging systems to RPC. The increasing adoption of SOAP on the Internet will enable a new generation of Web Services to be developed. Up until now Web Services have relied heavily on technologies such as CGI and Servlets. These take simple information in the form of text from a HTTP request, process it to provide a service and then return a response. The most familiar examples are Web pages that allow users to fill text into forms for submission to the server. SOAP provides a much more powerful interface into such servers allowing users to make more complex method calls to invoke more sophisticated services. This allows SOAP to be used in a large variety of systems ranging from messaging systems to RPC.

SOAP consists of three parts:

1. The SOAP envelope construct defines an overall framework for expressing what is in a message, who should deal with it, and whether it is optional or mandatory.
2. The SOAP encoding rules defines a serialization mechanism that can be used to exchange instances of application-defined datatypes.
3. The SOAP RPC representation defines a convention that can be used to represent remote procedure calls and responses.

Although these parts are described together as part of SOAP, they are functionally orthogonal. In particular, the envelope and the encoding rules are defined in different namespaces in order to promote simplicity through modularity.

### 2.6.2.3 UDDI

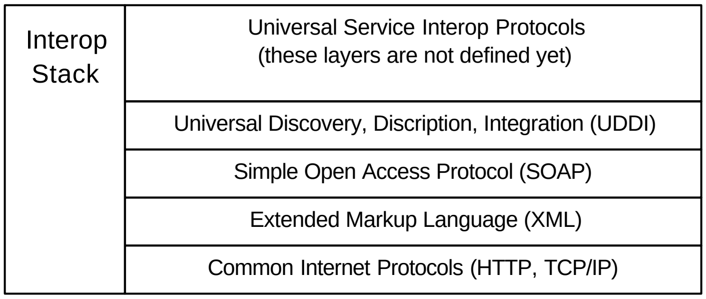
The Universal Discovery Description and Integration (UDDI) specifications define a way to publish and discover information about Web Services. The term "web service" describes specific business functionality exposed by a company, usually through an Internet connection, for the purpose of providing a way for another company or software program to use the service.

At first glance, it would seem simple to manage the process of Web Service *discovery*. After all, if a known business partner has a known electronic commerce gateway, what's left to discover? The tacit assumption, however, is that all of the information is already known. When you want to find out which business partners have which services, the ability to discover the answers can quickly become difficult. One option is to call each partner on the phone, and then try to find the right person to talk with. For a business that is exposing Web Services, having to staff enough highly technical people to satisfy random discovery demand is difficult to justify.

Another way to solve this problem is through an approach that uses a Web Services description file on each company's web site. After all, web crawlers work by accessing a registered URL and are able to discover and index text found on nests of web pages. The "robots.txt" approach, however, is dependent on the ability for a crawler to locate each web site and the location of the service description file on that website. This distributed approach is potentially scalable but lacks a mechanism to insure consistency in service description formats and for the easy tracking of changes as they occur.

UDDI takes an approach that relies upon a distributed registry of businesses and their service descriptions implemented in a common XML format. UDDI specifically consist of an XML schema for SOAP messages, and a description of the UDDI API specification. Together, these form a base information model and interaction framework that provides the ability to publish information about a broad array of Web Services. The UDDI specifications borrow the lesson learned from XML and SOAP to define a next-layer-up that lets two companies share a way to query each other's capabilities and to describe their own capabilities.

The following diagram depicts this layered view:



Interop Stack

Universal Service Interop Protocols
(these layers are not defined yet)

Universal Discovery, Discription, Integration (UDDI)

Simple Open Access Protocol (SOAP)

Extended Markup Language (XML)

Common Internet Protocols (HTTP, TCP/IP)

UDDI is a "next layer" in an emerging *stack* enabling rich Web Services. UDDI uses standards-based technologies such as TCP/IP, HTTP, XML and SOAP to create a uniform service description format and service discovery protocol.

Refer to the following URL for detailed information pertaining to UDDI

http://uddi.microsoft.com/developer/tech_white_paper.doc

### 2.6.3 Portal Integration Standards and Practices

The following sections describe the portal standards that must be adhered to by all integrating applications. Most importantly, the response generated by the application/service shall conform to DoD Section 508, “Web page accessibility” and all security policies stated by Department of the Navy.

### 2.6.3.1 Incorporate Portal Templates with Module Server

This section illustrates issues to be considered by the Service Developers for the incorporation of portal-defined templates and styles. The portal uses different style sheets for each template or theme; however, the name of the style sheets and their elements (referred to as “tags”) remain the same.

It is advisable that the Service Developers incorporate these cascading style sheets (CSS) into their applications to keep the look and feel across all applications consistent with the portal and to create a more seamless, user-friendly experience. In some instances, maintaining a template’s predefined color palette may be critical for a particular working environment, such as a ship’s command center where the implemented template may be designed for a dark room environment and a bright white application would be hard to use. Incorporating the CSS will promote display consistency across multiple pages and once incorporated, will save time in both developing and maintaining existing and new applications.

How to incorporate the Portal Template into Service:

1. Include the URL path used to reference the Cascading Style Sheets
2. Include tags (selectors/elements) in applications to define the attributes to be implemented

- **Include the URL path used to reference the Cascading Style Sheets**

The Service Developer must reference the style sheets at the top of their web pages, between the “header tags”.

For example:

```
<HEAD>

<LINK REL='stylesheet' HREF='/servlet/media/templates/’ & <ClientStyle> & ‘/styles.css'
TYPE='text/css' title=’TEMP_STYLES’>

</HEAD>
```

The above referenced style tag will be an attribute defined in the PRIDataRequest as ClientStyle. If the Service Developer chooses to incorporate the user’s look and feel into their application, the line above will need to be included in each page. (See Table 7-1)

- **Include tags in applications to define the attributes to be implemented**

In addition to referencing the user’s specific CSS, the Service Developer will also need to reference each style tag (Elements) as defined in the CSS.

For instance, below is HTML that may be in the Service Developer’s current web application:

```
<Table>

    <tr>

<td><b><font size=”10” type=”Arial” color=”blue”>Welcome, John Doe!</font></b></td>

    </tr>

</Table>
```

Assuming the CSS is referenced at the top of the page, the above would be replaced with:

```
<td class=”welcome”>Welcome, John Doe!</td>
```

where CSS tag “welcome” is defined in the stylesheet as:

```
.welcome
{
   COLOR: blue;
   FONT-FAMILY: arial, verdana, helvetica;
   FONT-WEIGHT: bold;
   FONT-SIZE: 10px
}
```

The value in using the CSS, is that if changes to the rendering are needed, e.g. fonts, colors, margins, typefaces and other aspects of style, on a web application, only one change is made in the CSS, rather than in all pages that use these styles.

How to reference specific the style tags defined by the CSS:

Below are a table and screenshots that demonstrate how to implement the style tags in the CSS. The table lists all of the elements and classes as defined by all style sheets used in the portal. The attributes for these elements and classes will change depending on the template chosen, however the code will not need to be modified once classes are referenced. The attributes listed below are one example of a template available on the portal. The screenshots map out where these have been used in a template for the portal. These may be used as a guideline for Service Developers to reference when adding the class names to their web applications.

**Table 3-5: Style Tag Descriptions for Style Sheets**

| **Element/Class** | **Description** | **Attributes** | **How to reference** |
|---|---|---|---|
| A | Hyperlink | COLOR: #0163e4;<br>FONT: 10pt univers, verdana, arial, helvetica, sans-serif;<br>TEXT-DECORATION: none | No additional code needed |
| Td | Table Data | FONT: 8pt univers, verdana, arial, helvetica, sans-serif | No additional code needed |
| **Th** | **Table Header** | FONT: bold 10pt univers, verdana, arial, helvetica, sans-serif | No additional code needed |
| contentheader | Header | COLOR: black;<br>FONT-FAMILY: univers, verdana, arial, helvetica;<br>FONT-SIZE: 12px | class=”contentheader” |
| currentdirectory | | COLOR: #a9a9a9 | |
| explorerbg | Background Color | BACKGROUND-COLOR: #8CAAE7 | class=”explorerbg” |
| explorertabindicator | | FONT-FAMILY: univers, verdana, arial, helvetica;<br>FONT-SIZE: 12px;<br>TEXT-DECORATION: none | class=”explorertabindicator” |
| explorertablebg | | BACKGROUND-COLOR: #f3f3f3 | class=”explorertablebg” |
| file | File font | COLOR: darkblue;<br>FONT-FAMILY: univers, verdana, arial, helvetica;<br>FONT-SIZE: 11px | class=”file” |
| fileselected | Selected File | BACKGROUND-COLOR: #FF9933;COLOR: #ffffff;<br>FONT-FAMILY: univers, verdana, | class=”fileselected” |

| Element/Class | Description | Attributes | How to reference |
|---|---|---|---|
| | | arial, helvetica;<br>FONT-SIZE: 11px | |
| folder | Folder Name | COLOR: #666699;<br>FONT-FAMILY: univers, verdana, arial, helvetica;<br>FONT-SIZE: 11px;<br>FONT-WEIGHT: bold | class="folder" |
| folderselected | Selected Folder | BACKGROUND-COLOR: #FF9933; | COLOR: #ffffff; |
| font1 | Font option | FONT: 12pt univers, verdana, arial, helvetica, sans-serif | class="font1" |
| font2 | Font option | COLOR: #919191;<br>FONT: 14pt univers, verdana, arial, helvetica, sans-serif | class="font2" |
| font3 | Font option | FONT: 8pt univers, verdana, arial, helvetica, sans-serif | class="font3" |
| libraryselected | Selected Library | BACKGROUND-COLOR: #FF9933; | COLOR: #ffffff |
| librarypath | Background color option | BACKGROUND-COLOR: #f3f3f3 | class="librarypath" |
| lightwash | Background color option | BACKGROUND-COLOR: #f3f3f3 | class="lightwash" |
| mediumwash | Background color option | BACKGROUND-COLOR: #f3f3f3 | class="mediumwash" |
| menuitem | Menu Items | COLOR: white;<br>FONT-FAMILY: univers, verdana, arial, helvetica;<br>FONT-SIZE: 11px;<br>FONT-WEIGHT: bold | class="menuitem" |
| menulink | Menu Link | FONT: 14pt univers, verdana, arial, helvetica, sans-serif | class="menulink" |
| message | | BACKGROUND-COLOR: ##EFEFEF;<br>COLOR: black;<br>FONT-FAMILY: univers, verdana, arial, helvetica;<br>FONT-SIZE: 16px | class="message" |
| mout | Mouse Out | COLOR: darkblue | class="mout" |
| mover | Mouse Over | COLOR: red | class="mover" |
| na | | COLOR: #2a71ac;<br>FONT: bold 10pt univers, verdana, arial, helvetica, sans-serif | class="na" |
| nc1 | | BACKGROUND-COLOR: #c9e6ff | class="nc1" |
| nc2 | | BACKGROUND-COLOR: #f3f3f3 | class="nc2" |
| nh | | COLOR: #919191;<br>FONT: 15pt univers, verdana, arial, helvetica, sans-serif | class="nh" |
| notselected | | BACKGROUND-COLOR: #ffffff;<br>COLOR: #385273 | class="notselected" |
| selected | Selected Option | BACKGROUND-COLOR: #ffffff;<br>COLOR: #666699;<br>FONT-FAMILY: univers, verdana, arial, helvetica;<br>FONT-SIZE: 11px;<br>FONT-WEIGHT: bold | class="selected" |
| title | Title | COLOR: #333366;FONT: 18pt univers, verdana, arial, helvetica, sans-serif | class="title" |

| **Element/Class** | **Description** | **Attributes** | **How to reference** |
|---|---|---|---|
| toolbar | Toolbar | BACKGROUND-COLOR: #6BA8E6 | class=”toolbar” |
| upload | | BACKGROUND-IMAGE: url(/servlet/media/images/base/toolback.gif);<br>VERTICAL-ALIGN: top | class=”upload” |
| white | | COLOR: #ffffff | class=”white” |
| wpadvice | Large Instructions | COLOR: #555555;<br>FONT-FAMILY: univers, verdana, helvetica;FONT-SIZE: 24px;<br>FONT-WEIGHT: bold | class=”wpadvice” |
| wpcontentlist1 | | BORDER-BOTTOM: #6666CC;<br>BORDER-LEFT: #6666CC;<br>BORDER-RIGHT: #6666CC;<br>BORDER-TOP: ##EFEFEF | class=”wpcontentlist1” |
| wpcontentlist2 | | BORDER-BOTTOM: ##EFEFEF;<br>BORDER-LEFT: ##EFEFEF;<br>BORDER-RIGHT: ##EFEFEF;<br>BORDER-TOP: ##EFEFEF | class=”wpcontentlist2” |
| Wpdefaultcursor | Default cursor style | CURSOR: default | class=”wpdefaultcursor” |
| wpelemtoolbar | | COLOR: #ffffff;<br>FONT-FAMILY: univers, verdana,arial,sans-serif;<br>FONT-SIZE: 8pt;<br>FONT-WEIGHT: bold | class=”wpelemtoolbar” |
| wpoptions | Options | FONT-FAMILY: univers, verdana,arial,sans-serif;<br>FONT-SIZE: 8pt;<br>FONT-WEIGHT: normal;<br>TEXT-DECORATION: none | class=”wpoptions” |
| **wpselectedtitle** | **Selected title** | BACKGROUND-COLOR: #8CAAE7;<br>COLOR: #ffffff;<br>FONT-FAMILY: Arial, Helvetica, sans-serif;<br>FONT-SIZE: 9pt;<br>FONT-WEIGHT: bold;<br>TEXT-DECORATION: none | class=”wpselectedtitle” |
| wptitle | | BACKGROUND-COLOR: #e5eaee;<br>FONT-FAMILY: univers, verdana,arial,sans-serif;<br>FONT-SIZE: 8pt;<br>FONT-WEIGHT: normal;<br>TEXT-DECORATION: none | class=”wptitle” |
| **wptoolbar** | **Toolbar** | BACKGROUND-COLOR: #e5eaee;<br>FONT-SIZE: 9pt;<br>FONT-WEIGHT: bold | class=”wptoolbar” |
| wptreetop | Background image | BACKGROUND-COLOR: ##EFEFEF;<br>BACKGROUND-IMAGE: url(/servlet/media/templates/16/images/background.gif);<br>COLOR: white | class=”wptreetop” |

### 2.6.3.2 Portal Friendly Service Development

Service Developers *may* need to modify existing application code to have the service work properly in the TFWeb Portal. They may need to modify the application code to ensure the service works appropriately when accessed via Reverse Proxy from within the Portal Framework.

The Portal Connector reverse proxy feature handles the reconfiguration and rewriting of links to properly flow back through the portal infrastructure, but there may be potential reverse proxy issues with “absolute paths” versus “relative paths”. Some development may be necessary if the paths are generated dynamically or programmatically.

These service development recommendations can be considered by the Service Developers to ensure a portal friendly interface between their application and the TFWeb Portal.

### 2.6.3.3 Reverse Proxy

When the Portal Connector is used to proxy access to URIs in web content, it does so by re-writing the links to redirect connections back through the portal. This allows a single access point through the portal through any firewalls, and ensures that content is managed through the portal interface. The Portal Connector examines the HTML on a web page and looks for certain key tags. When it encounters one of these tags, it prepends a call to the Portal Connector to the URI. When the Portal Connector gets a call of this type, it sets up an HTTP client session and requests the content on behalf of the user. The content is then examined for URIs to re-write and forwarded on to the user.

While this is a powerful capability, it does require that application/service developers be aware of certain limitations.

1. The Portal connector must be able to identify the link to re-write it. The connector identifies the following HTML tags for re-writing:

HREF=

SRC=

URL=

BACKGROUND=

ACTION=

All other methods for producing links, especially those that rely on client side code or code imbedded in objects is not supported and will result in an application being considered as “Level 1” integration. The filter cannot handle links it can't find to re-write.

2. It is good design practice to use relative links within HTML for specifying some links. This means that the object being referred to is at a location relative to the page being displayed. For example:

```
<img src="images/mygif.gif">
```

refers to a graphics image in a folder one level below where the HTML is located. In many cases, the Portal Connector can accurately rewrite these references. To do so, it must be able to establish the "base" URL. This can be determined in most cases for “Level 2” integrated applications as the header contains a document reference that can be prepended to the relative link. For “Level 3” integration, the Portal Connector has no idea of the URI to the content that is presented, so the use of the header tag "BASE HREF=" is required in the HTML header:

```
<HEAD> <TITLE>Page Title</TITLE> <BASE
HREF="http://homeport.nmci.navy.mil/html/"> </HEAD>
```

This allows the Portal Connector to establish the URL base as defined in:

- Section 12 of the HTML 4.1 standard (http://www.w3.org/TR/html4/)

- RFC 1808 Relative Uniform Resource Locators
- RFC 2616 Hypertext Transfer Protocol -- HTTP/1.1

As defined in the HTML standard, an undefined or underivable base will result in an unresolvable URI and a "broken" link.

For example: if the folder structure on this site looked like the following:

```
http://homeport.nmci.navy.mil
/---   html   /  --- images
```

and the current document resides in the html folder,relative links should look like this:

```
<img src="images/mygif.gif"> <a href="nextpage.html">Next Page</a>
```

When rewritten by the Portal Connector, the links would look something like this:

```
<img src= "http://portal/PortalConnector/user=joe@http://homeport.nmci.navy.mil/
html/images/mygif.gif">
<a href= "http://portal/PortalConnector/user=joe@http://homeport.nmci.navy.mil
/html/nextpage.html">Next Page</a>
```

### Impact of Reverse Proxy by Integration Level

Hyperlink Integration (Level 1)

The Portal Connector affects no HTML in “Level 1” applications as all content, other than the initial hyperlink, occurs in communication directly between the client browser and the application/web server.

Presentation Integration (Level 2)

All HTML in “Level 2” applications is run through the Portal Connector. The base URL is derived from the path to the page in which the link is embedded. If an application developer suspects that the proxy may not be able to detect the proper base, the base should be explicitly defined using the <BASE HREF=tag>, as defined above. URIs generated on the client side cannot be proxied under any circumstances.

Application/Data Integration (Level 3)

URIs produced by “Level 3” Service Modules are required to explicitly state the base URL. (see above). URIs generated on the client side cannot be proxied under any circumstances.

## 2.6.4 Development Coding Standards and Practices

In order to maintain a large organization of applications and services, certain code based naming conventions need to be applied. This guide is presented as a straightforward suggestion that will streamline potential conflicts within each service and application. This is only meant to be a guideline where there are no guidelines present. Where there are current guidelines, those should take precedence over any procedures in this guide. Many of these suggestions have been adapted as the “industry standard” or “best of breed” and are well known within the IT industry.

Questions and suggestions should be referred to the Open Source Site at https://tfw-opensource.spawar.navy.mil/RegRepTeamApps/WebHelp/

### 2.6.4.1 Directory Structure and Variable Naming Conventions

While the directory structure for a service module is highly subjective to the internal plans of the particular development shop, it was felt necessary to provide some guidelines on directory structure and variable management. Below is a suggested outline for application directories. In the next section variable management will be discussed. This should not supersede any internal mandate already in place.

The use of this outline is suggested and voluntary. This section was written to aid in the development process for the service application provider

### Assumptions

- The web administrator or web master has provided space for the web site on the web server
- All virtual directories have been created and configured by the web master or web administrator on the web server
- The user has sufficient rights to add, edit and delete files and directories in the target environment
- A saved backup incase a restore is required
- A clear idea of the functionality that is to be represented by the web site
- Source code control procedures or applications are being used
- All URL references should use relative URL instead or exact URL references.

## 2.6.4.2 Directory Structure for Storing Services in the Enterprise Module Server

The TFWeb Enterprise Module Server (EMS) provides support for Service Modules developed in J2EE, ASP or CGI, where CGI refers to Windows-compliant C++ or Perl. The location on the EMS file system where the Service Module is stored is based on the Service Module type (ASP, JSP or CGI). The location and type of Service Module also affects the URL that the portal connector uses to address the Service Module.

The following sub-sections outline where on the EMS file system to install each type of service module.

### 2.6.4.2.1 J2EE Service Modules

BEA WebLogic Server (WLS) 6.1 is the execution engine for J2EE Service Modules. J2EE Service Modules are deployed in BEA WLS as Web Applications or Enterprise Applications, with the distinction between each being the type and number of J2EE components being deployed.

### J2EE Web Applications

A J2EE Web Application contains the following types of resources:

- Servlets
- Java Server Pages (JSP)
- JSP Tag Libraries
- Static HTML pages and images

Although limited to containing only these types of resources, a Web Application is still able to access all services and APIs available in WLS, including EJB components, JDBC database connections and Java Messaging Service (JMS) resources.

Web Applications components are packaged in a Web Archive (WAR) file, which is a Java Archive (JAR) file, with a .war extension. WAR files bundle all component files in a directory into a single file, maintaining the directory structure. WAR files also include XML descriptors that instruct WLS how to deploy the components.

The J2EE Web Application WAR file must packaged in compliance with J2EE standards, as described at http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/WCC3.html, and must follow the naming standards defined J2EE Naming Standards Section

### J2EE Enterprise Applications

An Enterprise Application may contain a larger set of components, including:

- Web Applications (one or more)
- Enterprise Java Bean (EJB) components[1]
- Connector components – resource adapters

A J2EE Enterprise Application, consisting of assembled Web application(s), EJB components, and resource adapters, is packaged as an Enterprise Application Archive (EAR) file, which is a JAR file with an .ear extension.  Web Applications are packaged in a WAR file.  Enterprise Java Beans are packaged in JAR files with .jar extensions.  Resource adapters are packaged in a JAR file with a .rar extension.  An .EAR file contains all of the .jar, .war, and .rar component archive files for an application and an XML descriptor that describes the bundled components.

Each Web Application contained within a J2EE Enterprise Application file corresponds to a service module, must be assigned a service key, and must comply with all J2EE Web Application requirements.

The J2EE Enterprise Application EAR file must packaged in compliance with J2EE standards, as described at http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/Overview4.html, and must follow the naming standards defined in Section 0.

### J2EE Addressing Standards

J2EE service modules are addressed in the following manner:

https://<ems-hostname>/servlets/<service-key>/<entry-point>.<ext>

Where:

- `<ems-hostname>` is the fully qualified domain name of the EMS server
- `<service-key>` is the 32-character Globally Unique Identified (GUID) assigned to the service when it is registered in the Service Registry
- `<entry-point>` is the filename that corresponds to the entry point to the service
- `<ext>` is the appropriate extension of the service module entry point

The following is an example based on the above:

https://services.homeport.navy.mil/servlets/ACC7A3AB-B29C-47FE-A300-D7DE965FC530/myService.jsp

### J2EE Directory Standards

J2EE Service Modules are stored within the EMS as BEA WLS applications.  BEA WLS applications must be stored in the following directory:

```
<BEA-Product-Directory>\config\<BEA-Domain>\applications
```

Where:

- `<BEA-Product-Directory>` is the directory identified during installation where BEA product files will be installed (i.e. e:\bea\wlserver6.1)

[1] The Enterprise Module Server v1.1 supports only stateless EJB components.

- `<BEA-Domain>` is the BEA domain name identified during installation (i.e. EMSDomain)

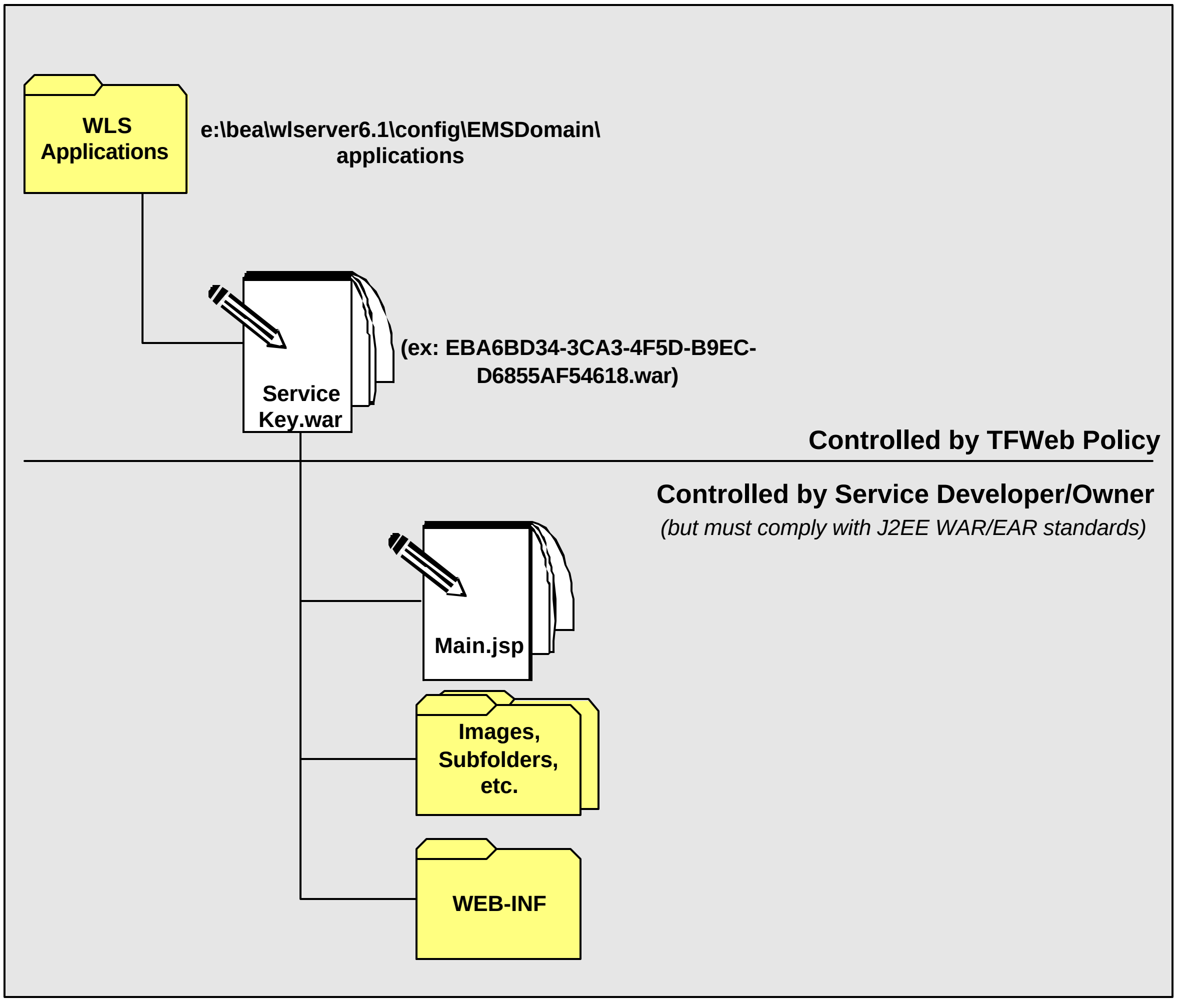

WLS Applications
e:\bea\wlserver6.1\config\EMSDomain\ applications
Service Key.war
(ex: EBA6BD34-3CA3-4F5D-B9EC-D6855AF54618.war)
Controlled by TFWeb Policy
Controlled by Service Developer/Owner
(but must comply with J2EE WAR/EAR standards)
Main.jsp
Images, Subfolders, etc.
WEB-INF

**Figure 0-1J2EE Directory Structure**

## J2EE Naming Standards

J2EE Service Modules are stored on the EMS as either Web Archive (WAR) or Enterprise Application Archive (EAR) files. The filename for the service must follow the following convention:

```
<service-key>.<ext>
```

Where:

- `<service-key>` is the 32-character Globally Unique Identifier (GUID) assigned to the service when it is registered in the Service Registry
- `<ext>` is the appropriate file extension of the application archive (i.e. .war or .ear)

The following is an example:

```
ACC7A3AB-B29C-47FE-A300-D7DE965FC530.war
```

In the case of EAR files, which may contain more than one Web Application, each Web Application stored within the EAR must be assigned (and named after) a unique service key.  The EAR file will be named using the service key of the primary Web Application stored within the EAR.

For example: ACC7A3AB-B29C-47FE-A300-D7DE965FC530.ear.

### 2.6.4.2.2 ASP Service Modules

Microsoft IIS is the execution engine for Active Server Pages (ASP) Service Modules.  Microsoft ASP is a server-side scripting language that allows for the creation of dynamic Web content.  An ASP Service Module may also contain static HTML pages and images.

ASP Service Modules are delivered to TFWeb in Microsoft CAB archive format.  The service modules are then un-archived and stored on the EMS server in exploded directory format in order to be accessed.

#### ASP Addressing Standards

ASP Service Modules are addressed in the following manner:

https://<ems-hostname>/<service-key>/<entry-point>.asp

Where:

- `<ems-hostname>` is the fully qualified domain name of the EMS server
- `<service-key>` is the 32-character Globally Unique Identified (GUID) assigned to the service when it is registered in the Service Registry
- `<entry-point>` is the filename that corresponds to the entry point to the service

The following is an example of an ASP URL:

```
https://services.homeport.navy.mil/ACC7A3AB-B29C-47FE-A300-
D7DE965FC530/myService.asp
```

#### ASP Directory Standards

ASP Service Modules are stored within the EMS as Microsoft IIS applications, which require that a new virtual directory be created for each Service Module.  ASP services are stored under the IIS standard *inetpub* directory, in a sub-directory that corresponds to the assigned service key.

```
<wwwroot>\<service-key>
```

where:

- `<wwwroot>` is the default IIS directory for storing web applications (e.g. e:\inetpub\wwwroot)
- `<service-key>` is the 128- bit Globally Unique Identified (GUID) assigned to the service when it is registered in the Service Registry

An example would be:

```
E:\inetpub\wwwroot\ACC7A3AB-B29C-47FE-A300-D7DE965FC530
```

After installing the service module, a new virtual directory must be configured that corresponds to that directory.  Virtual directories are created via Wizard from the Microsoft Internet Services Manager tool.  The service key should be used as the alias for the new virtual directory.  Virtual directories created for ASP service modules will require “Read” and “Run scripts” security permissions (configured during virtual directory creation).

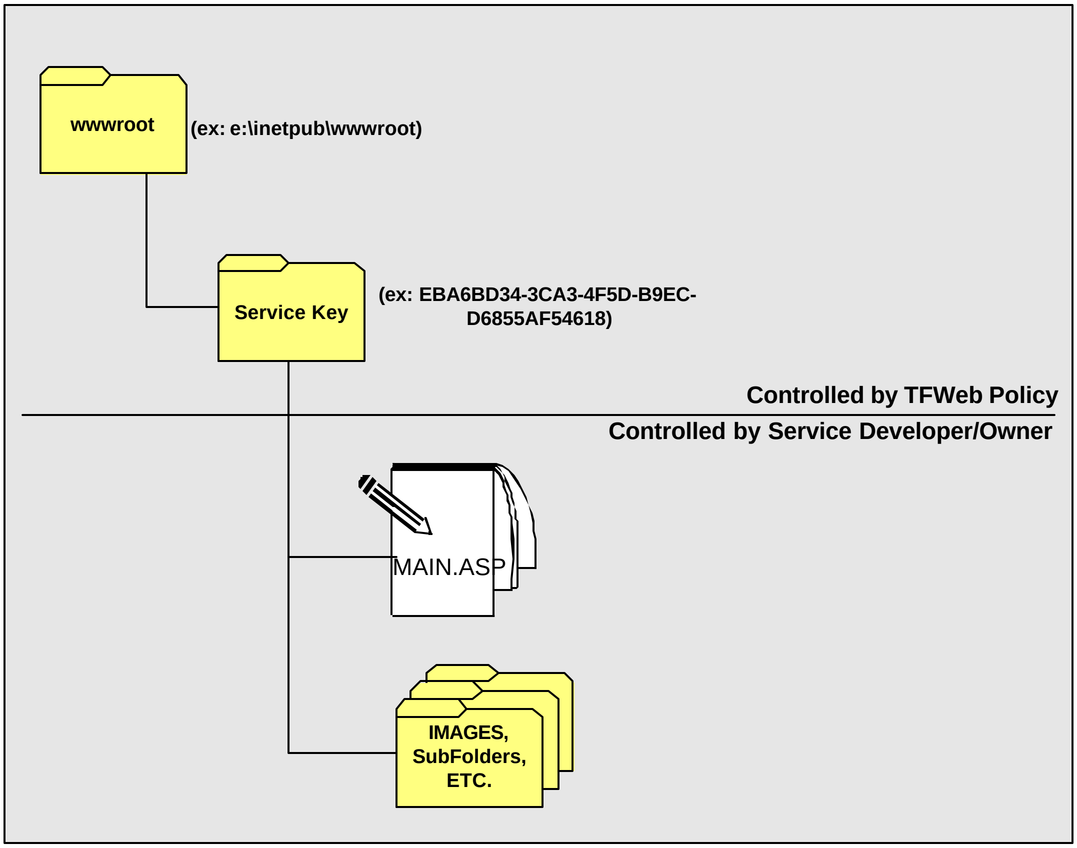

wwwroot
(ex: e:\inetpub\wwwroot)
Service Key
(ex: EBA6BD34-3CA3-4F5D-B9EC-D6855AF54618)
Controlled by TFWeb Policy
Controlled by Service Developer/Owner
MAIN.ASP
IMAGES, SubFolders, ETC.

**Figure 0-2ASP Directory Standards**

### ASP Naming Standards

Naming standards for ASP Service Module content stored under the service key directory is at the discretion of the service developer/owner, with the following exceptions:

- Files containing ASP scripts shall end in the extension .asp.
- Files containing HTML content shall end is the extension .html or .htm
- Files containing XSL content shall end in .xsl

The use of common naming standards is recommended as best practice for Web development, but is not provided as policy within this document.

### 2.6.4.2.3 Static HTML Service Modules

The EMS can also support Service Modules that contain only static HTML pages and associated images content, in particular for Level 1 (Hyperlink) Integration Service Modules. Static HTML Service Modules shall be stored and addressed using Microsoft IIS in the same manner as ASP Service Modules, with one exception – virtual directories created for static HTML Service Modules will require only “Read” security permissions (configured during virtual directory creation).

The following is a sample URL to access a static HTML service module:

```
https://services.homeport.navy.mil/ACC7A3AB-B29C-47FE-A300-
D7DE965FC530/myService.html
```

### 2.6.4.2.4 CGI Service Modules

CGI Service Modules consist of either Perl or Windows-compliant C/C++. ActiveState ActivePerl 5.1.6 is the execution engine for Service Modules developed using Perl. C/C++ is not an interpreted language, and hence does not require an execution engine. It is executed natively by the Windows operating system.

### CGI Addressing Standards

CGI Service Modules are addressed in the following manner:

```
https://<ems-hostname>/<service-key>/<entry-point>.<ext>
```

Where:

- `<ems-hostname>` is the fully qualified domain name of the EMS server
- `<service-key>` is the 32-character Globally Unique Identified (GUID) assigned to the service when it is registered in the Service Registry
- `<entry-point>` is the filename that corresponds to the entry point to the service
- `<ext>` is the appropriate extension for the service module type (i.e. .pl for Perl and .exe for C/C++)

The following is an example of a CGI URL:

```
https://services.homeport.navy.mil/ACC7A3AB-B29C-47FE-A300-
D7DE965FC530/myService.pl
```

### CGI Directory Standards

CGI Service Modules are stored within the EMS as Microsoft IIS applications, which require that a new virtual directory be created for each Service Module. CGI services are stored under the IIS standard *wwwroot* directory, in a sub-directory that corresponds to the assigned service key.

```
<wwwroot>/<service-key>
```

Where:

- `<wwwroot>` is the default IIS directory for storing web applications (e.g. e:\inetpub\wwwroot)
- `<service-key>` is the 32 character Globally Unique Identified (GUID) assigned to the service when it is registered in the Service Registry

An example would be:

```
E:\INETPUB\WWWROOT\ACC7A3AB-B29C-47FE-A300-D7DE965FC530
```

After installing the service module, a new virtual directory must be configured that corresponds to that directory. Virtual directories are created via Wizard from the Microsoft Internet Services Manager tool. The service key should be used as the alias for the new virtual directory. Virtual directories created for ASP service modules will require “Read”, “Run scripts” and “Execute” security permissions (configured during virtual directory creation).

### CGI Naming Standards

Naming standards for CGI Service Module content stored under the service key directory is at the discretion of the service developer/owner, with the following exceptions:

- Files containing Perl scripts shall end in the extension .pl
- Files containing Windows-compliant C++ code must be compiled and end in the extension .exe
- Files containing HTML content shall end is the extension .html or .htm
- Files containing XSL content shall end in .xsl

The use of common naming standards is recommended as best practice for Web development, but is not provided as policy within this document.

#### 2.6.4.2.5 Filename Standards

General filename standards should also be present when developing web sites.  Each file should tell a little bit about what the file should do.  This helps developers to organize code in a way that is logical and somewhat organized.  As always the shop rules apply to filename naming conventions before applying any outside rules.

Each file should be saved in its appropriate directory, with the appropriate extension, in order to promote organization and reuse.

### 2.6.4.3 Variable Management

Variable management is also another aspect of web site planning that is highly subjective to each shop.  If the current shop has already published guidelines for variable management, all parts of this discussion that comply with the policy should be followed, noncompliant directions should not be followed.  This discussion is for service developer’s information and should be taken into account when using client side variables.

Each service module is created within its own virtual directory, therefore, each service module will run within its own address space and there will be no collisions between applications for variable names.  Also, because each service module will be running within its own address space the use of global.asa with ASPs is allowed.

By complying with the broad guidelines below, “variable collision” could be held to a minimum. “Variable collision” can be defined as two variables with the same name that have different functionality within applications.  The collision occurs when the variable is called and the desired functionality does not occur, other functionality has rendered the variable inconsistent with the desired results.  This behavior can occur when using cookies, JavaScript or other client side validation techniques.

#### 2.6.4.3.1 General Naming Conventions for Variables

In general naming conventions should be meaningful to the web site developer and should describe the functionality of that specific variable.  As with any programming language, any variables should be named to express function or purpose.  Care should also be taken to not use reserve words as variables because there could be unexpected results.  When appropriate the developer should comment the application to aid with maintenance issues.

Comments help to explain why and how this part of the code works.  This allows for more detailed documentation right where the developer needs it, in the code.  There are many different ways to comment code a standard should be defined and followed throughout the coding effort. Check with the particular programming language to detail how to comment functionality within the code.

### 2.6.4.3.2 Local Variables

Local variables are variables that reside inside a function or procedure. These variables should not have subsequent pages rely on the values, as they will disappear on any subsequent page. Nonetheless, local variable naming should also express function or purpose. When necessary, it is always a good idea to type cast and declare variables (dim persarray(9) as array, declare persarray[] as array).

The following are examples of good variable names.

- personcount – counter to increment number of people logged in.
- lname - last name

Some examples of inefficient variable names:

- Ddrfvdse -unless it makes sense
- Yadayadayada – not descriptive enough
-

### 2.6.4.3.3 Global Variables

Global variables should be avoided if possible. If a global variable is used, make sure the variable is prefaced with some indicator that it is globally unique. A good naming standard is one that is planned in advance. This will also aid in the "non-collisionary" variable path that each web site seeks to encounter. Once a global variable is not used, destroy it so as not to encumber other application specific functions.

The following are some examples of global variables:

- gblUserID – the gbl designates that the UserID is global
- gv_Role – gv_ designates that the Role is a global variable
- globalRank – global designates the scope of the variable

Some poorly defined global variables:

- Out – could be confused with other functions like print.out
- In – could be confused with other functions like input()
- Count – reserved word
- Id- could be confused with any other id that may be used

### 2.6.4.3.4 Cookies

- **Temporary Client Side Cookies**

Temporary cookies are allowed. These are cookies that are removed from the browser when the web application ends. These cookies can maintain temporary pieces of information that are needed during the execution of the service module or backend application.

- **Permanent Client Side Cookies**

The use of permanent client side cookies should be restricted as these cookies can always be tracked back to users or user computers.   DoN CIO has found that cookies are in violation of a federal policy that prohibits the use of Internet technology that collects identifying information on individuals who access its web sites. That policy prohibits the use of web technology to collect identifying information to build profiles on individuals, and prohibits the use of persistent cookies unless certain conditions are met, including obtaining the personal approval of the head of the agency.

If a cookie must be used, use the GUID number. This a unique number for each web site on the NMCI web portal.  This unique naming convention will almost guarantee that cookies are not over written.

### 2.6.4.3.5 Server Side Session Variables

Server side session variables should use the same naming convention as all other variables.  As always the shop conventions should be adhered to before changing any parts of the code.  Session side variables should be used sparingly as they take up memory on the server and could potentially cause lags in service.

In order to save some of the processing power, be sure to destroy all unused session variables at the time the session variable is no longer used.

### 2.6.4.3.6 JSP, CGI and ASP Standards

Refer to the following web sites for the most up to date information regarding standards for the various development languages.

All languages TFWeb Open Source Site:

https://tfw-opensource.spawar.navy.mil/RegRepTeamApps/WebHelp/

ASP: http://msdn.microsoft.com/library/default.asp?url=/nhp/Default.asp?contentid=28000522

CGI: http://msdn.microsoft.com/

Cleartrust:  http://www.rsasecurity.com/products/cleartrust/index.html

BEA Documentation can be found at the following web site: http://e-docs.bea.com/wls/docs61/index.html.  Documents from the BEA site that are will help with the development and deployment of services on BEA site are:

- Programming WebLogic Enterprise JavaBeans at: http://e-docs.bea.com/wls/docs61/ejb/index.html
- Programming WebLogic JSP at: http://e-docs.bea.com/wls/docs61/jsp/index.html
- Assembling and Configuring Web Applications at: http://e-docs.bea.com/wls/docs61/webapp/index.html

IIS reference at web site: http://www.microsoft.com/

CleverPath (formerly Jasmine):  http://ca.com/products/jasmine/app_server.htm

Java Coding standards please follow the standards listed at web site:

http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html

JSP: http://java.sun.com/products/jsp/

### 2.6.4.3.7 Environment Cleanup

Environment cleanup refers to cleaning up variables, record sets, objects, connections, and streams after you are done with them. As each object is no longer being used, it is a good idea to destroy these objects to save memory leaks and to have the application perform at an optimum level. Do not rely on the garbage collector to clean up the environment. It is up to each developer to make sure that his or her environment is optimal.

### 2.6.4.4 Informaiton Assurance Standards and Practice

All information pertaining to IA is expressed in Section 3 of this document.

## 2.6.5 Emerging Standards and Practices for Reference

The Navy has identified some emergent technologies associated with web enablement for possible future standards that may be implemented into the Navy Service Oriented Architecture.

# 2.6.5.1 WSDL 1.1

WSDL defines an XML-based grammar for describing network services as a set of endpoints that accept messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly. They are bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services). WSDL is extensible to allow the description of endpoints and their messages regardless of what message formats or network protocols are being used to communicate. However, the only bindings described in this document describe how to use WSDL in conjunction with SOAP 1.1, HTTP GET/POST, and MIME.

- **Specification**

http://www.w3.org/TR/wsdl

- **Schema**

WSDL Framework: http://schemas.xmlsoap.org/wsdl/.

WSDL SOAP binding: http://schemas.xmlsoap.org/wsdl/soap/.

WSDL HTTP GET & POST binding: http://schemas.xmlsoap.org/wsdl/http/.

WSDL MIME binding: http://schemas.xmlsoap.org/wsdl/mime/.

- **Status**

WSDL 1.1 was submitted to the W3C and became a W3C Note 15 March 2001.

- **Introduction**

SOAP defines a message as an Envelope and allows users to define specific Headers and Body formats using XML. XML Schemas (XSD) provides a mechanism for describing an XML format, but cannot describe a message or endpoint. The Web Services Description Language (WSDL) is an XML-based document format that introduces an extensible grammar for describing message endpoints while leveraging XSD for defining message content.

- **Goals**

1. Transport and encoding extensibility: New transports and encodings can be added to the base specification without having to revise it.
2. Abstract definitions: Endpoints and messages can be described abstractly, and then mapped onto one or more concrete transports or encodings.
3. Reuse of definitions: Existing endpoint definitions can be used to create new definitions.

- **Non-goals**

1. Flow language: WSDL describes four basic message flow patterns (one-way, request-response, solicit-response, and notification) and leaves description of more complex flows to other specifications that extend the base patterns.
2. Expose implementation details: WSDL focuses on describing wire formats, not on describing implementation details of an endpoint.
3. Exchange of documents: WSDL defines a document format for describing message endpoints but leaves the exchange of such documents to other specifications (such as UDDI).

- **Details**

1. The WSDL grammar contains the following elements that are used together to describe endpoints:
2. Message: References to XML Schemas defining the different parts of the
3. Operation: Lists the messages involved in one message flow of the endpoint. For example, a request-response operation would refer to two messages.
4. PortType: The set of message flows (operations) expected by a particular endpoint type, without any details relating to transport or encoding.
5. Binding: The transport and encoding particulars for a portType.
6. Port: The network address of an endpoint and the binding it adheres to.
7. Service: A collection of related endpoints.

- **Implications**

As communications protocols and message formats are standardized in the Web community, it becomes increasingly possible and important to be able to describe the communications in some structured way. WSDL addresses this need by defining an XML grammar for describing network services as collections of communication endpoints capable of exchanging messages. WSDL service definitions provide documentation for distributed systems and serve as a recipe for automating the details involved in applications communication.

- **Related Specifications**

1. WSDL builds on XML, XML namespaces, and XML Schemas.
2. WSDL is extensible, allowing other specifications that define new protocols to introduce WSDL-specific grammar for conveying information about those protocols.
3. WSDL deliberately does not define complex flow information, but rather leaves this to flow languages.

4. WSDL does not define how WSDL documents are exchanged, but instead leaves this to inspection and directory specifications such as UDDI.

References: This section has briefly discussed several technologies and protocols. Additional information on these topics can be found at the following web sites:

| | |
|---|---|
| • J2EE | http://java.sun.com/j2ee/ |
| • .NET | http://microsoft.com/net/ |
| • Web Services | |
| | i. http://webservices.org/ |
| | ii. http://www.xml.com/pub/a/2001/04/04/webservices |
| • UDDI | http://www.uddi.org/ |
| • XML | http://www.xml.org/ |
| • SOAP | http://www.w3c.org/tr/soap/ |

# 3.0 Integration Processes

Integration of an application in the WEN and NMCI requires application developers to complete several review and test processes. These processes are also highlighted in Appendix 9, ARG Checklists. Included are entry criteria for these processes and the rationale for exclusions/exceptions.

<INSERT SUMMARY OF NMCI PROCESS>

The TFWeb Service Certification Process is designed to ensure application services meet the security and functional standards of TFWeb and the Government prior to implementation within the production TFWeb Portal.

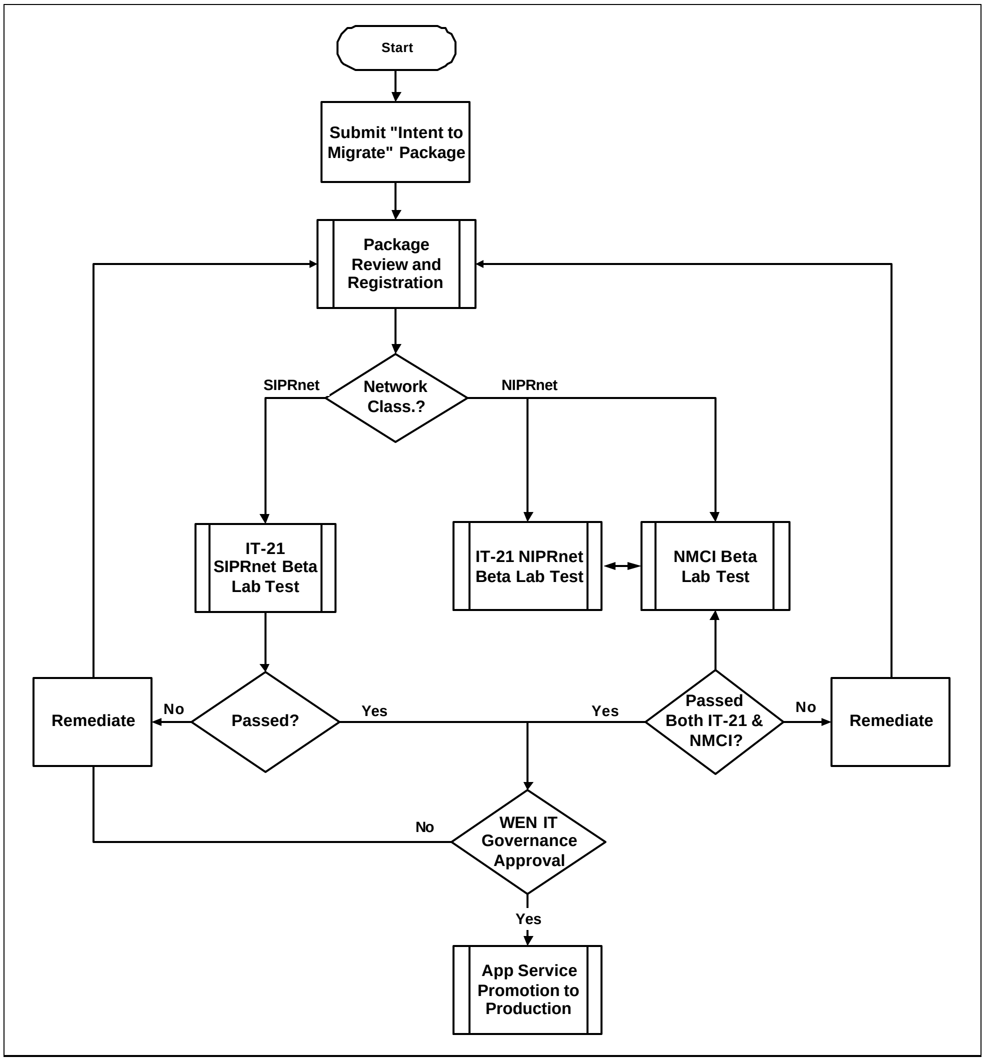

Start
Submit "Intent to Migrate" Package
Package Review and Registration
SIPRnet
Network Class.?
NIPRnet
IT-21 SIPRnet Beta Lab Test
IT-21 NIPRnet Beta Lab Test
NMCI Beta Lab Test
Remediate
No
Passed?
Yes
Yes
Passed Both IT-21 & NMCI?
No
Remediate
No
WEN IT Governance Approval
Yes
App Service Promotion to Production

**Figure 0-1 Service Certification Process.**

The certification process commences when the Service Owner delivers the Intent to Migrate Package to its Application Migration Customer Support (AMCS) point of contact (POC). The AMCS Team will review the Intent to Migrate Package for completeness and assists in assembling the required information for approval of the final Request to Migrate by the AMCS.

The TFWeb Beta Test Team will then perform Beta Testing within the IT-21 SIPRNET Lab for SIPRNET services and both the IT-21 NIPRNET and NMCI Beta Lab environments for NIPRNET services. The Test Team will communicate any issues encountered during testing to the Service Owner and the AMCS POC.

## 3.1 Application and Database Review

### 3.1.1 Data Collection and Assessment

NMCI has created the "ISF Tools" database which lists all current applications and database that reside on NMCI as well those in migration. This database also includes points of contact and status of the application through NMCI testing. All TFWeb applications must be listed in the database by June 1, 2002. Functional Application owners must provide developers of new or emerging applications the ability to review this data. It is the responsibility of both developers and application/database owners to conduct an analysis of this data to ensure that the proposed application does not duplicate existing functionality. Applications and databases with data overlapping the new application or database shall be identified and a data migration plan shall be established identifying the resolution of overlap. Approved resolutions are sharing data, establishing synchronization of data, or elimination of the legacy data source or program. Functional Application owners should ensure that shared access to any data generated and access to legacy data sources is established as a requirement from the beginning of application development. Also, owners as well as developers should ensure that their new application or database meets the technical standards of the WEN and NMCI early in the development process.

Since all applications are required to migrate to the TFWeb environment by 2004, all new applications and databases should submit an Intent to Migrate package to Task Force Web. This package consists of an entry in the application information database managed by TFWeb and a copy of the current IATO/ATO (if existing). The information will be reviewed by a member of TFW Application Migration Customer Support (AMCS) to determine the requirements for the final Service Registration package.

### 3.1.2 Rationalization

Rationalization is the process of identifying only those desktop and server-based applications, both COTS and GOTS, required to support command or DON missions, goals, and business processes. It includes the integration, consolidation, and elimination of applications and associated databases to improve standardization, enhance security, reduce duplication, and minimize support costs. Not all applications need be targeted for NMCI and TFWeb. Many application requirements will be met by utilizing alternative applications or by validating new requirements for applications under development. Rationalization policy and guidance is the responsibility of the DON CIO. Service-level policy and guidance for rationalization is the responsibility of respective Service CIO. Claimant/Marine Corps-level policy and guidance for rationalization of software applications is the responsibility of the CIO of the claimancy/Marine Corps organization. The DON CIO memorandum, "Management of DON Software Applications," dated 23 April 2001, promulgates policy for the review and reduction of applications and describes a plan for DON-wide application rationalization. This memorandum is available at: http://www.donimit.navy. mil/textversion/summaryTemplate.asp?catID=1&initID=44&theID=04262001OG A7754564. The DON-level rationalization process is a structured approach to an information management framework. This will include functional and acquisition program managers to ensure horizontal integration of systems and databases and will tie into the Enterprise Resource Planning and Task Force Web initiatives. This effort includes identifying duplicative applications, older versions of applications, applications that have already been certified, and others and working with the Navy claimants and Marine Corps organizations to resolve these issues.

### 3.1.3 Process for Rationalization

The purpose of rationalization is to reduce the number of redundant and/or obsolete applications and corresponding databases. It is the responsibility of the Functional Application Owners to ensure that the application has been rationalized in accordance with all applicable directives or guidance prior to the beginning of the development of a new or modified application.

The rationalization of GOTS applications begins with a search of the applications database to identify whether or not this application is a Department of Defense (DoD)-standard or DON-standard application, or whether your claimancy or Marine Corps organization, as appropriate, has accepted this application as a standard. If the GOTS application is flagged as a standard by any of these organizations, the rationalization process is complete and the application is moved to the next step in the transition process. Those GOTS applications that have not been flagged as standard are submitted to the Claimancy CIO, or USMC CIO, for review and approval. If the CIO approves this application as a standard, the CIO is responsible for submitting pertinent information for incorporation in the application database. The CIO shall consider Information Technology for the 21st Century (IT-21) and Marine Corps Tactical Network (MCTN) published standards as aids in their decision.

Some criteria that are normally disqualifying include the following: (WHERE ARE THESE FROM – some don't make sense for GOTS; more like COTS)

No personal, non-mission, or non-business-related software

No games

No freeware or shareware

No beta or test version software packages

No application development software (exception applies for approved science and technology [S&T] seats)

No agents

No duplication of standard seat services

No duplication of Contract Line Item Number (CLIN) 0023 software applications; CLIN 0023 applications are standard for their respective functional areas

Adequate business case for requirement must be demonstrated (NEED TO LINK TO AN EXAMPLE)

Applications must be compliant with DON/DoD Security policy (allowances should be made for applications mandated by other Government agencies that DON is required to use)

Exceptions to any of these rules must have the approval of either the claimant CIO or the Stakeholders' Council [SHC]).

All GOTS applications must receive the permission of their Echelon II CIO via the "ISF Tools" database prior to testing in the NMCI environment. In addition, all applications must be approved by TFWeb prior to testing in the NMCI or TFWeb environments.

The rationalization of COTS applications begins with a search of the NMCI contract, with amendments, to identify whether or not this version of this application is already included in the standard seat services or in CLIN 0023. If yes, then this version of this application is already approved for use on NMCI. If not, then using the applications taxonomy provided on the EDS NMCI Web site, look for and migrate to applications that offer duplicate functionality or an acceptable level of functionality and are included on the NMCI contract, with amendments. If an application is found with duplicate or similar functionality, but is not on the NMCI contract, then look for and migrate to applications that are certified for use on NMCI as identified on the NMCI PMO-certified applications list. See the NMCI PMO Web site for the latest list of approved, certified COTS applications. If there are no certified applications that provide duplicate or similar functionality, submit this fact to your claimant CIO or USMC CIO for review and approval as a standard for your community, using the same criteria listed in the GOTS rationalization process (above). With that approval, your application can be submitted via the NMCI PMO for certification testing. If the CIO approves this application as a standard, the CIO is responsible for submitting pertinent information for incorporation in the application database.

### 3.1.4 Task Force Web Registration

The Service Registration Package should be submitted in memo form by an authorized representative of the Echelon II command to the appropriate TFW AMCS POC via email and should address the issues listed below. Portions of existing documents may be submitted to prevent unnecessary duplication of effort. However, the submission should be organized to provide the following information. Guidance is provided in section 11.3 on the Service Registration Package review process

Interim Authority to Operate (IATO) or Authority to Operate (ATO) from the appropriate Designated Approval Authority (DAA) for the software developer.

NMCI Request for Service (RFS) form for NIPRNET applications

Migration plan to level 3 integration with appropriate milestones (separate document)

Waiver for level 1 integration included (if applicable)

Registry Metadata

Module Server package

Access control list (See Section 6.7.1)

Test plan and cases

Temporary login with access to non-administrator portions of the application - if a level 1 or 2 application

Summary of previous testing accomplished

Configuration of local application servers or remote module servers and estimated concurrent users of service

Documentation of application data structures and data interfaces

Migration plan - if application/data overlap has been identified

Migration plan - if XML not in compliance with Navy standards is in use

In some cases a single application will be comprised of multiple individual services, each with its own service module. If an application has multiple service modules then the service owner should submit a separate test plan and RFS (if applicable) as part of the migration package. Some services may also have multiple distributed physical instances. This typically requires each service instance to have a unique URL as well as a separate ACL. In this situation the service owner should submit a separate service module and ACL for each physical instance of the service with the migration package.

All contents of the Service Registration Package should be placed in a Zip file and forwarded to the AMCS POC via email.

### *3.1.4.1 Registry Metadata*

The Service Owner should include the following information for integration into the portal registry.

Description of portal service to be integrated

The URL of the service to be integrated

The Owner of the service to be integrated (lead development organization)

The taxonomy category under which the service will be listed (see Section 5)

Point(s) of contact information for user access. This information should include names, phone numbers, email addresses

Any parameter information required by the service

Target User Community (role/platform/location)

Service versioning information

### *3.1.4.2 Module Server Package*

The Service Owner will need to put together a module server package that includes all resources that the service will require for being included in the Enterprise Module Server. The module server package contents will vary depending on the level of integration required by the service.

These module server resources can include any of the following types of items: HTML pages, icons and images, XML files, XSL templates, JSP pages, Java Servlets, EJBs, ASP pages, COM/COM+ components.

The following file formats are acceptable for the module server packages:

Java developers should deliver their module server package in WAR/EAR format. WAR/EAR files are an executable file archive format used to package deployment files for a Java enabled application server.

ASP developers should deliver their module server package in CAB format. The CAB format is a file archive pattern used to package deployment files in the Microsoft Information Server environment.

### 3.1.5 Certification and Accreditation

Refer to DoD – DITSCAP tailorable; Navy IA Pub 5239-13 (vols I, II, & III).
The DoD Instruction (5200.48) Defense Information Technology Security Certification and Accreditation Process (DITSCAP) defines the activities leading to security C&A. Activities are grouped in a logical sequence. This instruction presents the objectives, activities, and management of the DITSCAP process. The objective of DITSCAP is to establish a DoD standard infrastructure-centric approach that protects and secures the entities composing the Defense Information Infrastructure (DII). The set of activities presented in DITSCAP standardize the C&A process for single IT entities that leads to more secure system operations and a more secure DII. The process considers the system mission, environment, and architecture while assessing the impact of operation of that system on the DII.

The Navy has documented implementation guidance for DITSCAP in Navy IA Pubs 5239-13 (vols I, II, & III). A main tenant of DITSCAP is tailorability. The level of effort (LOE) to accomplish C&A can be customized to the application seeking accreditation. DoN has based customization on application/system complexity, mission criticality, and the mode of operations of the environment that the application is functioning in. Detailed information can be found in the NMCI Connection Approval Process (NCAP).

It is the Service Owner responsibility to obtain an Approval to Operation (ATO) or Interim Approval to Operation (IATO) for an application prior to registering it for migration to the TFWeb Portal. Please see the Information System Security Manager (ISSM) representative for your command for more information.

### 3.1.6 Certification of Functional Need

Certification of Functional Need is the process by which the functional owner/milestone decision authority provides concurrence to the initiative.

### 3.1.7 Authoritative Data Source

*AUTHOR: AEAG*
*This section will describe how to identify the data owner (listing of Navy/Marine Corps data standards and sources (see section 2.2) and any associated processes. These are customer functions: when developers are subject to the authority herein and when the enterprise authority takes precedence over a Claimant Need; a catalog, listing of other approved applications; how to go about getting authority to modify an application belonging to another owner; how to go about making modifications to applications designed for joint use; how a developer determines the existing and approved applications available for use; and need to get questions answered prior to initiating any development actions.*

### 3.1.8 Application Hosting Determination Process

*AUTHOR: PMO*
This section intends to provide decision factors on the use of CLIN 0029 by the Acquirer of the Application, and to identify POCs needed for the described service.
Refer to
http://www.eds.com/nmci/clinlist.doc

#### 3.1.8.1 CLIN: 0029 Legacy Systems Support

**Service Description.** The Legacy Systems Support CLIN provides to the Acquirer of the Application the ability to obtain initial integration services for legacy applications as well as new or emerging operational and functional applications to enable them to run on NMCI. System support can also provide additional services beyond basic integration. These additional services provide a range of options that include, but are not limited to, NMCI ISF hosting of applications, operations and maintenance support, database management, and training, if ordered. This service may include participation of the NMCI ISF in business process re-engineering activities.

### 3.1.9 License Management

The ISF asset management scope includes software asset management for items procured by the ISF directly for, or in support of, a CLIN under the NMCI contract. Whether the DoN provides the ISF the 'right to use' or whether the ISF procures software to meet its own contractual obligations, the ISF will manage the licenses of that software, in accordance with the NMCI contract beginning with Section 1.0.

### 3.1.10 Approvals

*AUTHOR: PEO-IT*
*This section will describe any known DoN approvals required during the development and deployment of the application, including Claimant/Activity Application Review*

*Process guidance. This will describe how to identify the data owner and any associated processes: customer functions; when developers are subject to the authority herein and when the enterprise authority takes precedence over a Claimant.*
*A catalog listing of other approved applications; how to go about getting authority to modify an application belonging to another owner; how to go about making modifications to applications designed for joint use; how a developer determines the existing and approved applications available for use; and the need to get questions answered prior to initiating any development actions, all will be included. Parts are applicable to sections 1 and 6.*

## 3.2 Application Development

### 3.2.1 NMCI Development Environment

The Science and Technology (S&T) Working Group has defined CLINs to support the unique processing requirements of the S&T communities. These CLINs are numbered 0038AA-AH. Some of the requirements include:

- Ability to rapidly reconfigure hardware
- Ability to work collaboratively and share large data files
- Support for non-WIN2K Operating Systems
- Support for non-standard protocols
- High bandwidth requirements
- Appropriate security mechanisms

A detailed description of the CLINs can be found on the ISF web site at http://www.eds.com/nmci/catalog.htm.

### 3.2.2 Accreditation Plan

While developing the Systems Security Authorization Agreement (SSAA), one of the early activities is to develop the C&A strategy, plans, and LOE. This information is captured in the SSAA and agreed upon by the key C&A personnel (defined by DITSCAP as the DAA, CA, CA, ISSM, ISSOs, user reps, and the PM). The DITSCAP and Navy implementation documents describe the information required to develop the C&A Plan, LOE etc., and can be found at the Navy INFOSEC website URL: https://www.infosec.navy.mil. The specific NMCI C&A tailoring guidance can be found in the NCAP posted at http://www.eds.com/nmci

## 3.3 Before Visiting NMCI for an Engineering Review

The process of transitioning applications to NMCI entails a set of interrelated processes that impact various DoN components and the ISF. The Applications Resource Guide seeks to communicate the transition requirements and expectations with the objective of enabling the customer to effectively plan and efficiently execute their transition to NMCI.

### 3.3.1 Recommended Steps prior to an Engineering Review

The following checklist is recommended for use by developers prior to entering Engineering Review:

- Architecture Review Board Report
- Software Test Reports
- Code Review Inspection Reports
- Risk Management Plans
- Software Implementation Plan
- Software Users Manual or adequate Help Facility
- Configuration Management Plan
- Certification Accreditation Letters
- Software Quality Assurance Plan
- Release Procedures, if not included in the Implementation Plan
- A copy of the Engineering Review Question Set (provided by the ISF)
- A copy of the Security Working Group Process document.
- A copy of the Applications Resource Guide.

### 3.3.2 Security Certification and Accreditation Process

*AUTHOR: PMW161, MITNOC*

As described in VI.B.3 Security C&A, the C&A efforts integrated into the application should be appropriately documented in the SSAA Key elements of the SSAA for review are as follows:

- Definition and appointment of IA personnel (DAA, CA, CA, ISSM, ISSOS, user reps, and the program manager)
- Mission Description and System Identification
- Environment Description
- System Architectural Description
- System Security Requirements
- Organizations and Resources
- DITSCAP Plan

### 3.3.3 NMCI and Connection Approval Process (NCAP)

If application is accredited according to DITSCAP and Navy Policy, NCAP is a request for connection (RFC) process. RFC pulls the pertinent information from the application accreditation package to allow the NMCI connection decision authority (NMCI DAA) to make an informed connection decision.

If the C&A process has not been integrated, the NCAP defines the ways to tailor the DITSCAP to specific situations and still produce all necessary information to make a NMCI connection decision. The NCAP can be located at the Navy INFOSEC website at URL: https://www.infosec.navy.mil.

### 3.3.4 Testing Considerations

Applications must successfully complete the Developer Test and Evaluation (DT&E), including the creation of test scripts and testing scenarios. It must be verified that the application will work on an NMCI-certified workstation. Developers must describe the types of tests done in the NCMI Certification process (e.g., will the application print, will office applications continue to operate); any consideration for prototype/pilot testing; the steps, data, and logical conditions necessary to trigger programmed authentication processes (LDAP, Active Directory, file sharing, file writes, etc.) to ensure Group Policy, Lockdown, and Security areas are thoroughly examined by the Certification and Directory Services Teams. Developers ensure logon ids used have the same access rights as end-users, not developers.

## 3.4 TFWeb Beta Test Processes

TFWeb Beta Lab testing ensures that application services function appropriately within the portal environment and that they adhere to the TFWeb standards outlined in the Integration Developers Guide. There are three TFWeb Beta Test Labs, an IT-21 SIPRNET lab, an IT-21 NIPRNET lab, and an NMCI NIPRNET lab. All application services functioning across the SIPRNET will be tested in the IT-21 SIPRNET Lab. Services functioning across NIPRNET will be tested in the IT-21 NIPRNET Lab as well as the NMCI Beta Lab.

### 3.4.1 IT-21 NIPRNET Beta Lab Process

The AMCS POC will submit SIPRNET services packages the IT-21 SIPRNET Beta Lab. As packages are submitted for testing the following process is followed:

The AMCS POC submits a service package to Beta Lab POC via email. The package must include the following items required by the Beta Lab for testing:

Registry Meta-data

Module Server package

Test plan and test cases

Description of any special application functionality that will be required and/or tested

The Beta Test Team reviews the service package for completeness. If the package is complete, the service is scheduled for testing and notification is sent to the service owner and the AMCS POC. If the package is incomplete notification is sent requesting the missing components before scheduling the service for testing.

The Beta Test Team creates the global unique identifier (GUID) key for the service, installs the service module, creates the Registry entries, and creates the portal connector for the service.

The Beta Test Team performs the tests.

If the application fails any test cases or if its performance impacts that of the Portal environment the application will not pass the Beta Test. In this case the Service Owner and the AMCS

POC are notified with specific reasons for failure. AMCS may request support from AMTS to remediate any technical issues preventing approval.

Once the service has passed testing, the service will progress to the WEN IT Governance. Upon approval from WEN IT Governance the service is promoted to production.

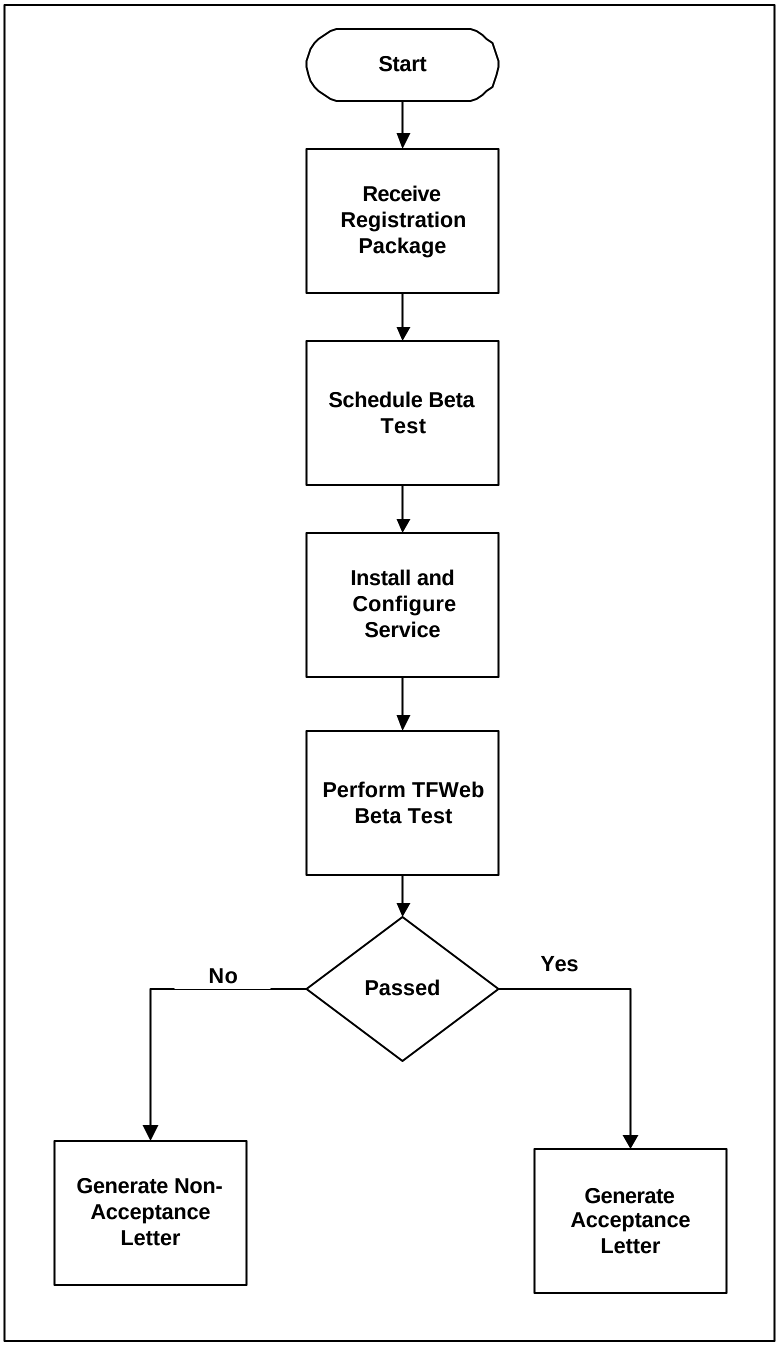

Start
Receive Registration Package
Schedule Beta Test
Install and Configure Service
Perform TFWeb Beta Test
Passed
No
Yes
Generate Non-Acceptance Letter
Generate Acceptance Letter

**Figure 0-2 TFWeb IT-21 SIPRNET Beta Test Process**

### 3.4.2 IT-21 NIPRNET Beta Lab Process

The AMCS POC will submit NIPRNET services packages to both the NMCI Beta Lab and the IT-21 NIPRNET Beta Lab. The labs coordinate so that each service package is tested in only one lab at a time. This helps balance the load between the labs and ensures that any issues encountered during testing are addressed by one lab thus minimizing delays in the other lab. As packages are submitted for testing the following process is followed:

The AMCS POC submits a service package to Beta Lab POC via email. The package must include the following items required by the Beta Lab for testing:

Registry Meta-data

Module Server package

Access control list

Test plan and test cases

Request for Service (RFS) form

Description of any special application functionality that will be required and/or tested

The Beta Test Team reviews the service package for completeness. If the package is complete, the service is scheduled for testing and notification is sent to the service owner and the AMCS POC. If the package is incomplete notification is sent requesting the missing components before scheduling the service for testing.

The Beta Test Team creates the global unique identifier (GUID) key for the service, installs the service module, creates the Registry entries, and creates the portal connector for the service.

The Beta Test Team performs the tests.

If the application fails any test case or if its performance impacts that of the Portal environment the application will not pass the Beta Test. In this case the Service Owner and the AMCS POC are notified with specific reasons for failure. AMCS may request support from AMTS to remediate any technical issues preventing approval.

The IT-21 Beta Lab collaborates closely on testing NIPRNET services with the NMCI Beta Lab. If the service has passed testing in the IT-21 NIPRNET Beta Lab it is then forwarded to the NMCI Beta Lab to complete the TFWeb certification process.

When the service passes the Beta Test in both the IT-21 and NMCI Beta Labs a notification letter will be sent to the Service Owner and the AMCS POC.

Once the service has passed testing, the service will progress to the WEN IT Governance. Upon approval from WEN IT Governance the service is promoted to production.

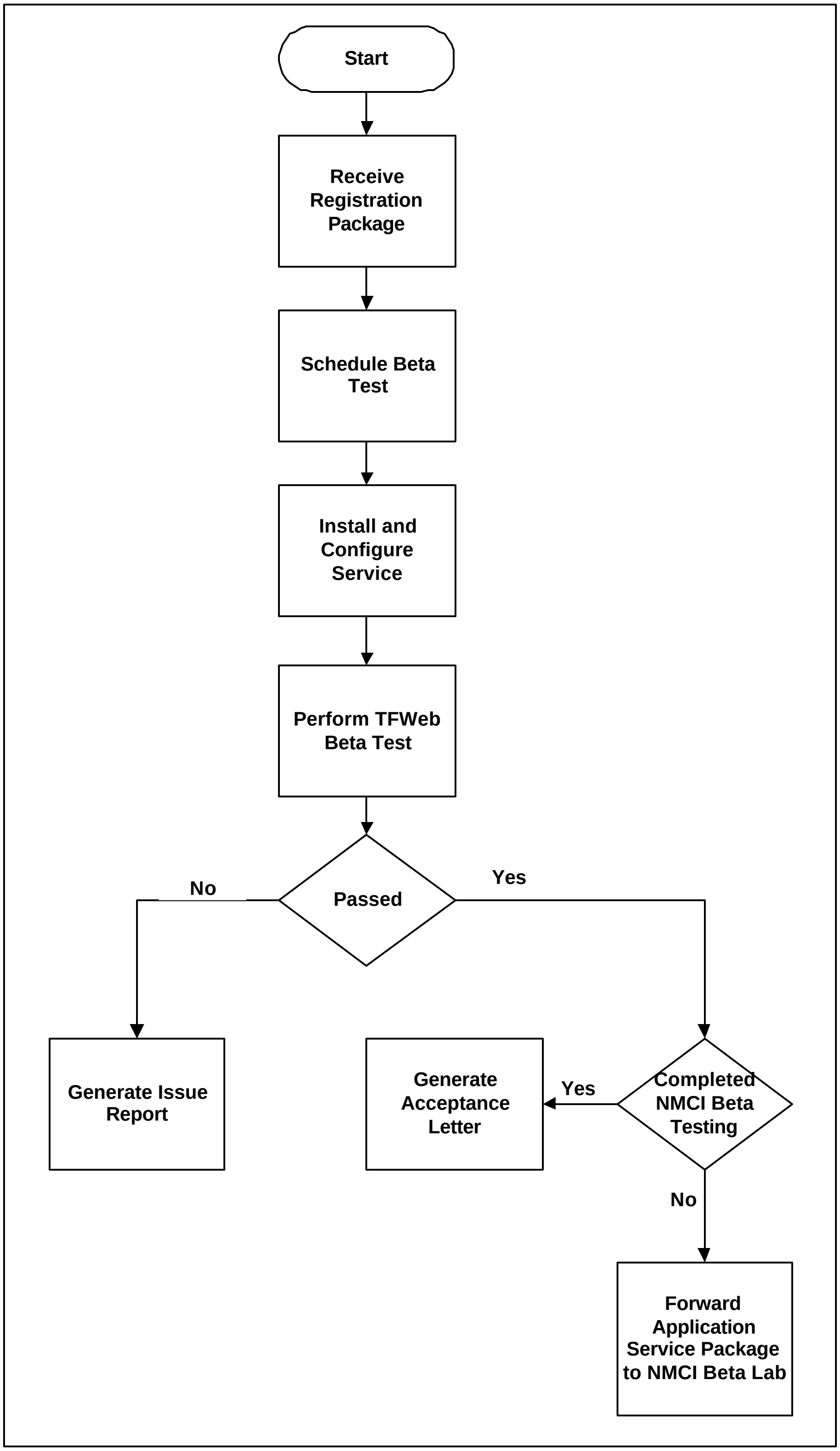

Start
Receive Registration Package
Schedule Beta Test
Install and Configure Service
Perform TFWeb Beta Test
Passed
No
Yes
Generate Issue Report
Generate Acceptance Letter
Yes
Completed NMCI Beta Testing
No
Forward Application Service Package to NMCI Beta Lab

**Figure 0-3 IT-21 NIPRNET Beta Test Process**

### 3.4.3 NMCI Beta Test Process

The AMCS POC will submit NIPRNET services packages to both the NMCI Beta Lab and the IT-21 NIPRNET Beta Lab. The labs coordinate tests so that each service package is tested in only one lab at a time. This helps balance the load between the labs and ensures that any issues encountered during testing are addressed by one lab thus minimizing delays in the other lab. As packages are submitted for testing the following process is followed:

The AMCS POC submits a service package to Beta Lab POC via email. The package will include the following items required by the Beta Lab for testing:

Registry Meta-data

Module Server package

Access control list

Test plan and test cases

Request for Service (RFS) form

Description of any special application functionality that will be required and/or tested

The Beta Test Team reviews the service package for completeness. If the package is complete, the service is scheduled for testing and notification is sent to the service owner and the AMCS POC. If the package is incomplete notification is sent requesting the missing components before scheduling the service for testing.

If a service requires the modification of the desktop configuration (i.e. plug-ins, active-X controls, etc.) then NMCI requires that desktop application go through an additional process in order to certify the security of the mobile code. NMCI also requires that the mobile code be tested on the standard NMCI desktop to ensure that it does not impact other standard desktop application. The latter process (NMCI Application Certification Process) is outlined in the following section.

The Beta Test Team creates the global unique identifier (GUID) key for the service, installs the service module, creates the Registry entries, and creates the portal connector for the service.

The Beta Test Team performs the tests.

If the application fails any test cases or if its performance impacts that of the Portal environment the application will not pass the Beta Test. In this case the Service Owner and the AMCS POC are notified with specific reasons for failure. AMCS may request support from AMTS to remediate any technical issues preventing approval.

A security (Green Team) scan is performed to ensure that the service module meets information assurance (IA) criteria.

The NMCI Beta Lab collaborates closely on testing NIPRNET services with the IT-21 NIPRNET Beta Lab. Services are not tested simultaneously in both labs. If the service has passed testing in the NMCI Beta Lab, but has not been tested in the IT-21 NIPRNET Beta Lab it is then forwarded to that lab to complete the TFWeb certification process.

When the service passes the Beta Test (both the IT-21 and NMCI Beta Labs for NIPRNET services) a notification letter will be sent to the Service Owner and the AMCS POC.

Once the service has passed testing, the service will progress to the WEN IT Governance. Upon approval from WEN IT Governance the service is promoted to production.

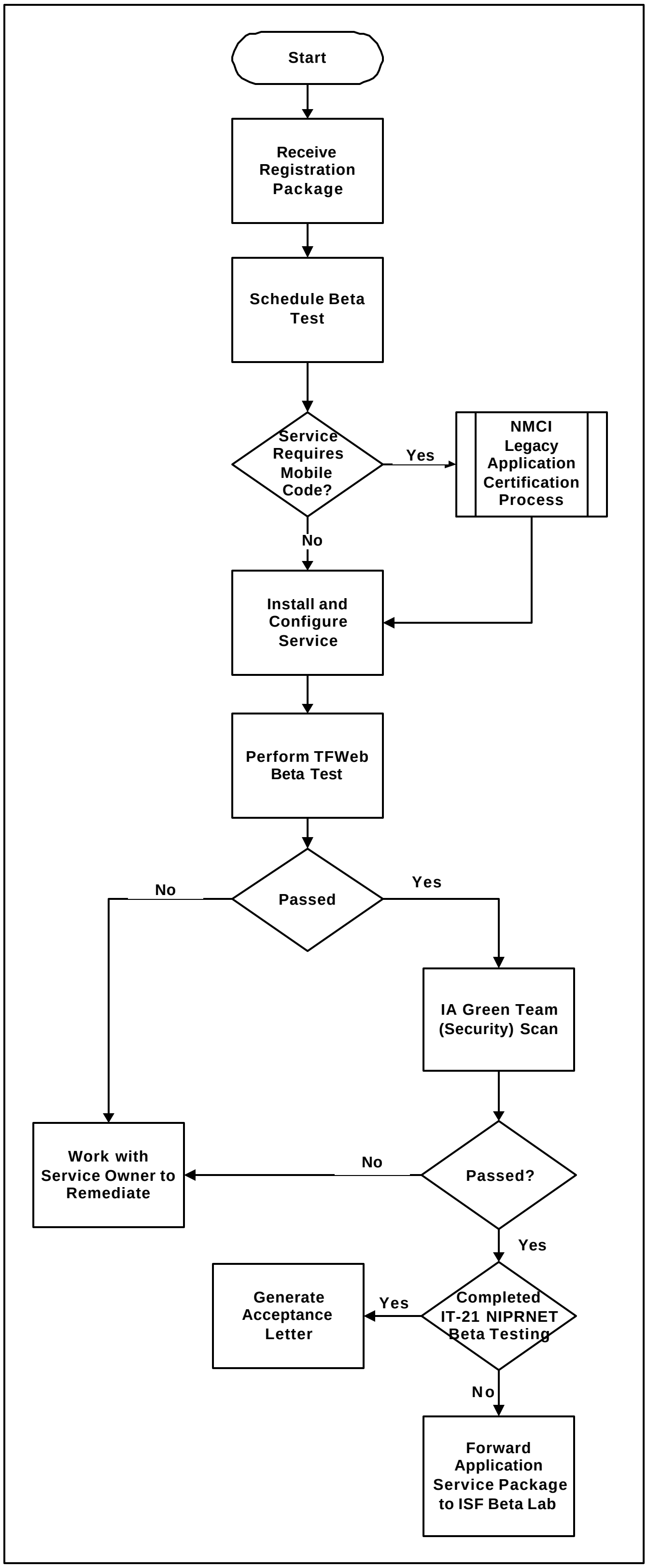

Start
Receive Registration Package
Schedule Beta Test
Service Requires Mobile Code?
Yes
NMCI Legacy Application Certification Process
No
Install and Configure Service
Perform TFWeb Beta Test
No
Passed
Yes
IA Green Team (Security) Scan
Work with Service Owner to Remediate
No
Passed?
Yes
Generate Acceptance Letter
Yes
Completed IT-21 NIPRNET Beta Testing
No
Forward Application Service Package to ISF Beta Lab

**Figure 0-4 NMCI Beta Test Process**

### 3.4.3.1 NMCI Application (Mobile Code) Certification Process

The NMCI Application Certification Team will be responsible for processing all desktop applications (e.g. mobile code) through a two-phase testing cycle. Phase 1 consists of basic application functionality testing, and Phase 2 consists of standard seat service integrity testing. Any special or additional test requirements must be identified by the Navy claimants prior to the beginning of testing, and preferably in the NMCI RFS.

The certification criteria currently required for Phase 1 testing include:

Basic functionality:

Launch the application

Create or Open a new document or file

Save a new document or file

Print a new document or file

Close the application

Execute "best business practice" or "customer-defined" testing scripts applicable to the application.

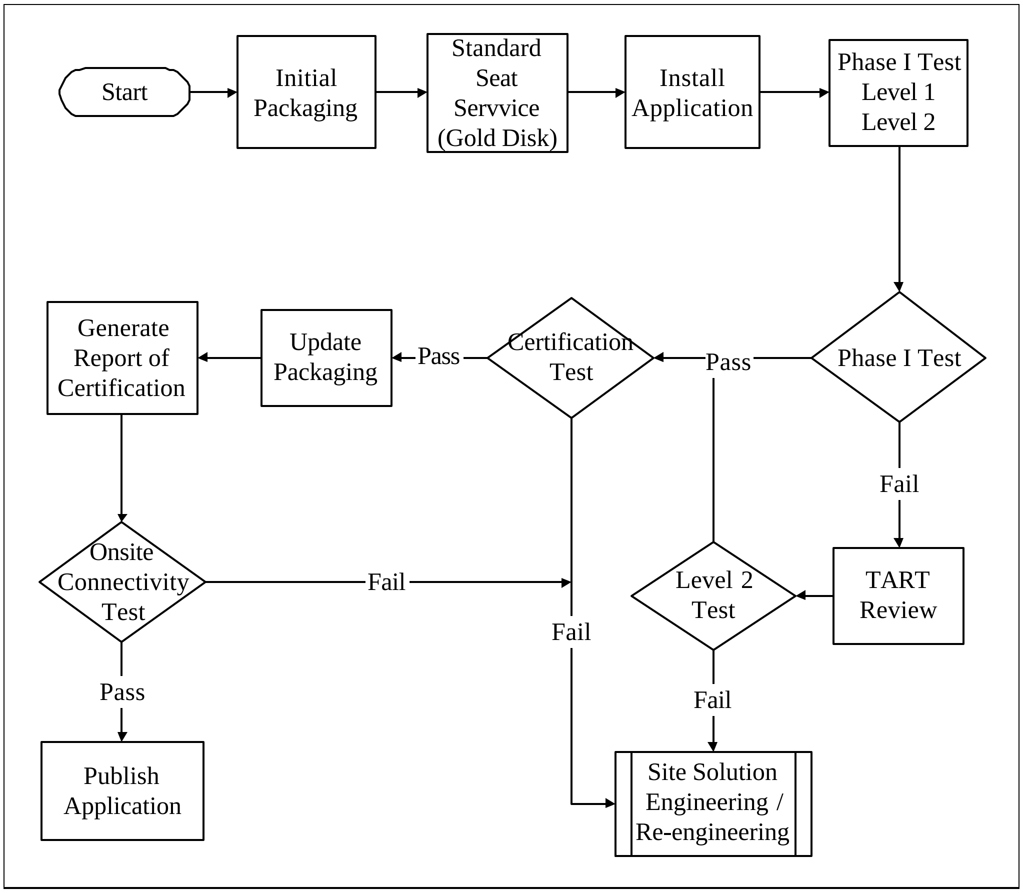


**Figure 0-5 NMCI Application Certification (Mobile Code) Process**

Any additional functionality testing needs to be defined and scripted by the Service Owner to ensure proper execution. Once the lab receives an application, the Service Owner can track the status by viewing the report posted online at http://www.eds.com/nmci/transition.htm.

## 3.5 Certification Lab Activity

For familiarization and preparation of the Application Certification Process, developers can initiate several processes and documents. The public NMCI web site, www.eds.com/nmci contains a link to a page titled Making the Transition'. This page has links to online documents for the following:

- Legacy Application Transition Guide
- Legacy Application Certification Liaison Letter (700-W02FN)
- Legacy Application Pre-Certification (700-W02FK)
- Legacy Application Certification - Request for Service (RFS) (700-W02FB)

For the purposes of this guide these documents can be used for either Legacy or New and Emerging Applications. Of these, the Transition Guide familiarizes developers with all the end-to-end processes for application transition into NMCI and the Liaison Letter serves as a checklist for preparation steps for the certification lab. An excerpt of the Liaison letter appears below.

### 3.5.1 Application Certification Liaison Letter

This letter describes the information and materials a site must submit to the NMCI Proving Center/Certification Lab (PCL) before the Lab can begin testing unclassified and classified applications for certification. If any media is received without an RFS then it will be considered Not Received (NR), or if an RFS is submitted without media then it will be considered Incomplete. In either case the site Government representative (CTR) and PMO will be contacted in order to acquire the appropriate documentation and/or media. All items will be tracked and pursued by the Certification Lab Site Liaison and PMO jointly in order to ensure completion of delivery. In order to maintain accountability for shipping and receiving every package should include a shipping list complete with content details (Application Name, Version, and clearly marked media). Upon receipt of the package, the Certification Lab will confirm with the site all applications received or not received to ensure accountability. Anything reported as sent but not received will be reported to the PMO and the site's CTR.

During testing an application's progress can be checked online at www.eds.com/nmci/transition.htm. Each application submitted to the Lab will complete the Application Certification Process, but before the Lab can begin testing, it must have sufficient information and materials.

### 3.5.2 Information/Materials for Lab Testing

The following are materials the laboratory must have for testing:

- A complete NMCI Request For Service (RFS).
- A valid key/license (if required).
- A copy of the application's original software media that is functional, readable, installable and complete.

- All available or applicable software documentation, including installation details and procedures.
- A description of any special application features and functions that will be required and/or tested, including server connectivity and access issues.
- Manual test scripts (step-by-step descriptions of test procedures) for special application functionality tests. The Lab may require a manual script to test a GOTS application or unusual software whose experienced users are not available for questions.

## 3.6 Certification Lab Process

This section describes the classified and unclassified lab certification processes, including POP in the Box (a mobile server that approximates the NMCI environment-permits testing to check configuration). Developers are encouraged to review the Certification Process documents and the NMCI Transition Guide to gain the full perspective of these processes. (See Appendix 10 and 17 for links to these documents.
The steps for Proving Certification Lab (PCL) processes are illustrated in Exhibit 11.

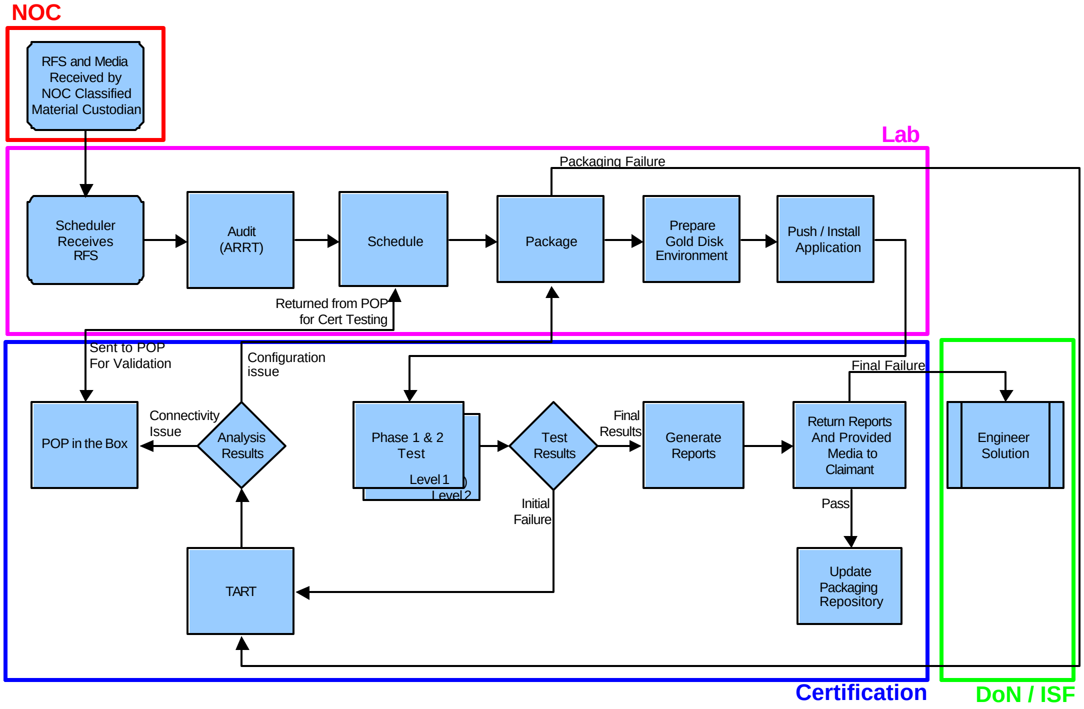


**Exhibit 11: Proving Certification Lab (PCL) Processes**

This process is applied identically for Classified and Unclassified applications with one exception. The first step initiated by the NOC: ‘RFS and Media Received by NOC Classified Material Custodian’ is only necessary for Classified applications.

### 3.6.1 PCL Process Steps

Customer/ISF Initiation - Initiation of the certification process:

- *'As Is' environment* – prior to cutover to NMCI, ISF, PMO and customer sites work together to identify and collect data on Legacy Applications, rationalize lists, then submit a Request for Service (RFS) for each Application.
- '*To Be' environment* – after cutover to NMCI, the Acquirer of the Application may introduce New or Emerging Applications by submitting a Request For Service via the proper chain of command and issuing a CLIN 0029 Task Order for certification testing.

Request For Service

- Will be the tool used to gather information from the Customer.
- This information will consist of Customer, Application, Installation, and Testing-specific information. In addition, the RFS should be accompanied with the appropriate media, key/license, and any Customer Test Scripts or Special Instructions, if applicable.

Audit (ARRT)

- A review process to assure that all informational and material requirements have been met for certification processing. Conducted internally at the Certification Lab by the Application Rationalization and Review Team (ARRT).

Scheduling

- After a successful audit the RFS is then scheduled to a resource/cell. If there is a need to prioritize a RFS, this should be done by contacting the PMO, who then conveys the priority need to the ISF/Certification Lab (PCL).

Packaging

- Is the process of combining an Application with automated installation scripts for use with the NMCI software distribution system (Novadigm Radia). The entire package must be certified.

Level 1 Testing

Level 1 constitutes actual Certification Testing and comprises two parts:

- Phase 1: Application Basic Functionality Testing – the application works after deployed to the NMCI environment.
- Phase 2: Gold Disk Integrity Testing – the application does not harm the NMCI environment.

Level 2 Testing (TART)

- Is only conducted on those Applications that fail Level 1 Testing, and in those cells identified for Advanced Application Certification Testing (AACT).
- AACT will be the process of redeploying the initial package to a specially equipped test cell that can provide a more detailed analysis of the application installation, configuration and packaging.

Validation

- Is conducted on site utilizing the PoP in the Box engineering tool. Some applications require site connectivity in order to validate application functionality and/or connectivity/security compliance.
- The Proving Center/Certification Lab may send an application to PoP in the Box pre or post testing in the Lab.

Certification Pass/Fail

- The responsible Certification Team Manager generates a NMCI Technical Certification Letter, NMCI Application Release Notes, and NMCI Certification Certificate stating the results of the Certification Process.

### 3.6.2 Parties to the Process

Following are parties to the PCL process:

- *Customer/Claimant* – The Navy and Marine Corps entity or site representative requesting the certification.
- *Application Owner* – The Navy/Government on site application administrator and/or user if he/she is both.
- *Central Development Activity (CDA)* – The Government application developer.
- *(Classified Applications only) NOC Classified Material (CMS) Custodian* – The ISF (Raytheon) individual responsible for receipt and accountability of classified material at the NOC facility.
- *Application Rationalization and Review Team (ARRT)* – This team is responsible for providing an initial review/audit of the RFS and ensuring all informational requirements have been fulfilled.
- *Lab Scheduler* – This is the individual responsible for managing the lab resources, and coordinating packaging and certification cells. Cell utilization and productivity will be the focus of this step.
- *Packaging Technical Lead* – The individual responsible for supervising the initial packaging team.
- *Certification Technical Lead* – The individual responsible for supervising the Testing Cycle and completing the NMCI Certification Technical Lead Checklist.
- *System Administrator* – The individual responsible for conducting the testing.
- *Technical Application Review Team (TART)* – This is the technical review team that will attempt to resolve installation or configuration issues that preclude an application from passing certification.
- *Certification Manager* – The manager responsible for the Certification Team that performed the testing.
- *Site Liaison* – Proving Center Lab personnel responsible for assisting, monitoring, and coordinating the application gathering effort.
- *POP in the Box* – This is an engineering tool that provides pre/post-validation of applications connectivity in order to certify for NMCI. It simulates the NMCI environment, and includes firewall, VPN, router, and client components.

- *Certification Data Warehouse (CDW)* – The database to be used to store, track, and control the certification process.

### 3.6.3 PCL Process Documents

- 700-W02FB NMCI Request For Service (Web based/Form)
- 700-W02FC NMCI Application Audit (Web based/Form)
- 700-W02FD NMCI Certification Technical Lead Checklist (Printed/Checklist)
- 700-W02FE1 NMCI Novadigm Radia Packaging Details (MSI) w/ Amendments (Web based/Form)
- 700-W02FE2 NMCI Novadigm Radia Packaging Details (Non-MSI) w/ Amendments (Web based/Form)
- 700-W02FF NMCI Application Installation Details w/ Amendments (Web based/Form)
- 700-W02FH NMCI Certification Test Checklist (Printed/Checklist)
- 700-W02FI NMCI Test Results Summary (Web based/Form)
- 700-W02FJ NMCI Technical Certification Letter (Web based/Report)
- 700-W02FK NMCI Application Release Notes (Web Based/Report)
- 700-W02FL NMCI AACT Details (Web based/Form)
- 700-W02FM NMCI Application Certification Certificate (Web Based/Report)
- 700-W02FN NMCI Application Certification Liaison Letter (Standard Letter)

### 3.6.4 Developer Impact

Developers perform the following:

- Required to follow the Certification processes and forms to have their application authorized to be operating within NMCI.
- Must follow these processes and related life cycle processes anytime application changes are performed and planned for release into NMCI.
- Be responsible for performing corrections and re-submitting the application for certification if lab results are unsatisfactory.
- Not required to be present (on location) at the Certification Lab during certification steps but are invited to do so if they wish.
- For POP in-Box testing, developers are responsible or involved in the Pre/Post Certification processes, documents, providing application test scripts, application installation instructions, user IDs, license keys, being present of installation (if necessary), etc.

## 3.7 Before Deployment/Migration

*AUTHOR: ISF / PMO*

### 3.7.1 WEN IT Governance

Programs that do not meet all requirements for migration may rarely be allowed to proceed through the testing process while simultaneously completing these requirements. In addition, applications that fail portions of the testing may be functionally displaced by another application by the time they are ready for migration to the production portal. Testing may also demonstrate substantial overlap with another application or organizational issues that prevent immediate migration of the application. Final approval of migration is currently a function of the Task Force Web Executive Steering Group. This approval may be delegated to a lower level based on application compliance with TFWeb standards.

### 3.7.2 Help Desk Procedures

Developers must ensure that application and NMCI help desk are properly notified and prepared to handle user issues. They link to any help desk processes from the transition guide. Developers add description of whether the help desk is being hosted by the ISF or identify who is providing the service; they need to update the desk providing the final service if partial help is provided from another source. See Appendix 16 for the NMCI Help Desk phone number.

### 3.7.3 Training

At the time of desktop installation, an initial, personal introduction to the machine is provided. In addition, extensive access to a variety of computer-based training courses also is available at no additional cost. SLA 17 defines training requirements.

### 3.7.4 Backup and Recovery

Developers must create and test an appropriate backup and recovery process and identify an up-to-date B/R plan.

## 3.8 Deployment/Migration

### 3.8.1 NMCI Hosting of Applications on Terminal Services

Many bases/sites/Commands have a pre-existing "thin client architecture" that serves as the foundation for how applications run and behave on a terminal server. Most of the server-based applications in the Navy/MC are based on the NT4 Terminal Server operating system. The existing Navy/MC architectures and assumptions are likely incompatible with the "NMCI Thin Client Architecture". For example, existing Navy/MC thin client architectures include security, permissions and domains standards that accommodate the applications. Moving the applications to the more stringent NMCI Windows 2000 infrastructure with new domains and security models makes it unlikely the applications will operate correctly without modifications. It is important to remember each base/site/Command may have their own “thin client architecture"; so leveraging solutions across sites/Commands/bases may not be possible.

### 3.8.1.1 Typical Scenarios for Hosting Applications on Terminal Services

There are four categories for moving/migrating/converting applications to a terminal server platform and three of them require issuing Task Orders under CLIN 0029 to host the application(s). The two high-level criteria for determining if CLIN 0029 needs to be executed are based on (a) leaving applications on existing platforms or (b) moving them to NMCI-supported hosted platforms:

**Legacy Application Access**: The claimant runs applications on Terminal Services today, and **the claimant wants to perform their own server support,** the ISF will provide connectivity to the "Legacy Application" through Terminal Services client(s). The claimant will maintain the servers and administration like other legacy applications. In this case, a software distribution package will be necessary to deploy the client software to the NMCI seat.

1. **Legacy Server Support**: If a claimant runs applications on Terminal Services today, and **they want the ISF to support pre-existing servers**, a Task Order under CLIN 0029 must be executed for re-engineering and hosting services.
2. **Move/Migrate/Convert Multi-User Legacy Application**: If a claimant runs applications on Terminal Services today, and **the claimant wants the ISF to engineer the applications to run on NMCI Terminal Servers,** a Task Order under CLIN 0029 must be executed for engineering and hosting services.
3. **Move/Migrate/Convert Single-User Legacy Application**: If a claimant does not use Terminal Services today, but **the claimant wants the ISF to engineer an application to run on ISF Terminal Servers,** CLIN 0029 Task Order must be executed for re-engineering and hosting services.

#### Results when Executing CLIN 0029 for Applications on Terminal Services

Determine compatibility with Windows 2000 Professional and Windows 2000 Terminal services.

4. Determine how many sessions a terminal server can support.
5. Determine reusability of existing hardware and software.
6. Determine network connectivity and Security requirements.
7. Determine ID, group and OU requirements
8. Determine if portal integration is necessary.
9. Determine performance measurements.
10. Determine ongoing costs, if any.

### 3.8.1.2 Programming Standards for a Terminal Server Platform

**Development Guidelines.** For applications to work well in a multi-user environment, certain programming standards must be used. Terminal servers host applications for multiple end-users, but the application must be written so that user-specific information is not tied directly to a machine. For example, applications cannot use the TCP/IP address to uniquely identify a user because many users on a terminal server share the same address. Microsoft provides guidance on the following categories:

- Building a Terminal-Services-Aware Application

- Application Setup in a Terminal Services Environment
- Storing User-Specific Information
- Kernel Object Name Spaces
- IP Addresses and Computer Names
- Client/Server Applications
- Graphic Effects
- Peripheral Hardware
- Background Tasks
- Thread Usage

See Appendix 3 for a link to the Microsoft site for Terminal Services Programming Guidelines.

**Tuning and Optimizing Applications.** In addition to the categories mentioned above, Microsoft provides specific tuning and optimization guidelines. Adhering to these standards helps ensure applications run efficiently, or run, or in some cases, run at all. The following standards are not only good to use for a multi-user platform, but are good best practice techniques. Programming guidelines to use are as follows:

- Support Customization Through User Profiles
- No Memory Leaks
- Do Not Replace System Files
- Do Not Assume Computer Name or IP Address Equates to Single User
- DCOM Support
- Consider the Peripheral Hardware Environment
- Do Not Assume Persistence of Files in Temp
- Disallowing Multiple Instances of Some Applications
- Do Not Assume the Windows Shell
- Do Not Modify the GINA
- Negotiate Client/Server Connections Inside the System and Network
- Multilingual and International Usage Scenarios

See Appendix 3 for a link to the Microsoft site for Optimizing Applications for Windows 2000 Terminal Services.

## 3.9 System Changes

Following are procedures for system changes.

### 3.9.1 Emergency Production Fixes

Emergency production fixes may be authorized only if the problem is critical or may jeopardize safety, or the problem adversely affects the mission and an interim workaround is not possible.

Emergency production fixes are not authorized if the following occur:

- The problem adversely affects the mission but a workaround may be used in the interim until the formal change process may be completed.

- The problem is inconvenient but does not affect essential capability.
- The change will adversely affect firewall policy compliance.
- Any question on the Recertification Checklist is answered “Yes”.

Procedures:

- The problem is investigated to determine the cause.
- A fix is developed.
- The fix is tested for adequacy.
- The fix is regression tested.
- The fix is entered into the configuration management process and tracked so that it can be entered into and follow the formal release process.
- The fix will be included in the next formal release.

### 3.9.2 Recertification Procedures

Once an application has been certified for NMCI under the application access process, any modifications to the application require re-certification. This re-certification effort, to include this distribution of the update, is a purchasable item from the contract. This orderable item is currently being developed (as of 08/22/01) and is anticipated to be available within the next month. It is currently not determined which CLIN will be used to make this service available for order. This document will be updated once the contractual activities have been completed. This CLIN would also be used for initial certification of "new" applications being introduced to NMCI.
Any code change will require re-certification. This includes hard-code logic changes, parameter changes in configuration files, include files, copybooks, etc., and any change that requires the application to be recompiled.

## 3.10 System Retirement/Sunset

*Processes and procedures for shutting down an application currently do not exist. Refer to the transition guide where appropriate. Should include the ISF and others impacted by the decision.*
Process for developers to follow when retiring a system under NMCI:

- Notify the users, NMCI, and any others of the application’s retirement date.
- Stop the application from running on the retirement date.
- Make a backup copy or and an archive to store for history purposes.
- Remove the application and any extra software needed to run from all applicable machines.
- Notify the users, NMCI, and any others that the system has been retired.

## 3.11 Reusable Components

Developers perform the following:

- Establish the procedures and tools to develop a reuse repository and associated policy.
- Establish a reuse component manager who will control the reuse component maintenance. This person will be responsible for approval, logging, and retirement of reusable components.
- Obtain a tool for tracking and logging of reusable components.
- Create a process for submitting and approval of reusable components.
- Keep track of a ranking and "lessons learned" history (a developers perspective) on reusable components.

## 3.12 Interfaces/Adapters

*Developers establish the procedures and tools to develop an interface adapters central repository and associated policy.*

## 3.13 Metrics

*Describe what reporting metrics that NMCI will provide the application owner. Describe what NMCI monitoring facilities are available for the application to communicate with. Describe what metrics are required by DoN / DoD to be tracked by applications. A developer's view…*
*Who are the peak users of my application?*
*How many are using my application?*
*What are peak usage times for the users of my application?*
*Visibility of any logs that may be generated when the application is run.*
*If the application has a problem, help with tracking and tracing the issue or bug.*

## 3.14 Knowledge Services

*Describe what is available from the knowledge management community to the application developer. Describe any associated procedures. Describe the developer news groups available to share information.*

## 3.15 Timelines (Generic)

The NMCI Transition Guide and Lab Certification processes (see Appendices) detail the requirements, processes, and general timelines. These documents contain specific and general time frames for all phases, from Data Gathering to Deployment. Because the rate of an application progressing through the transition processes will depend upon its network complexity (local area, intranet, or internet connectivity), business criticality (mission support / administrative or mission critical), and the parties involved, timelines for applications fully completing NMCI deployment will vary.
It is recommended for developers to review this Guide and the Lab Certification process to become fully familiarized with the necessary processes and time line guidelines.

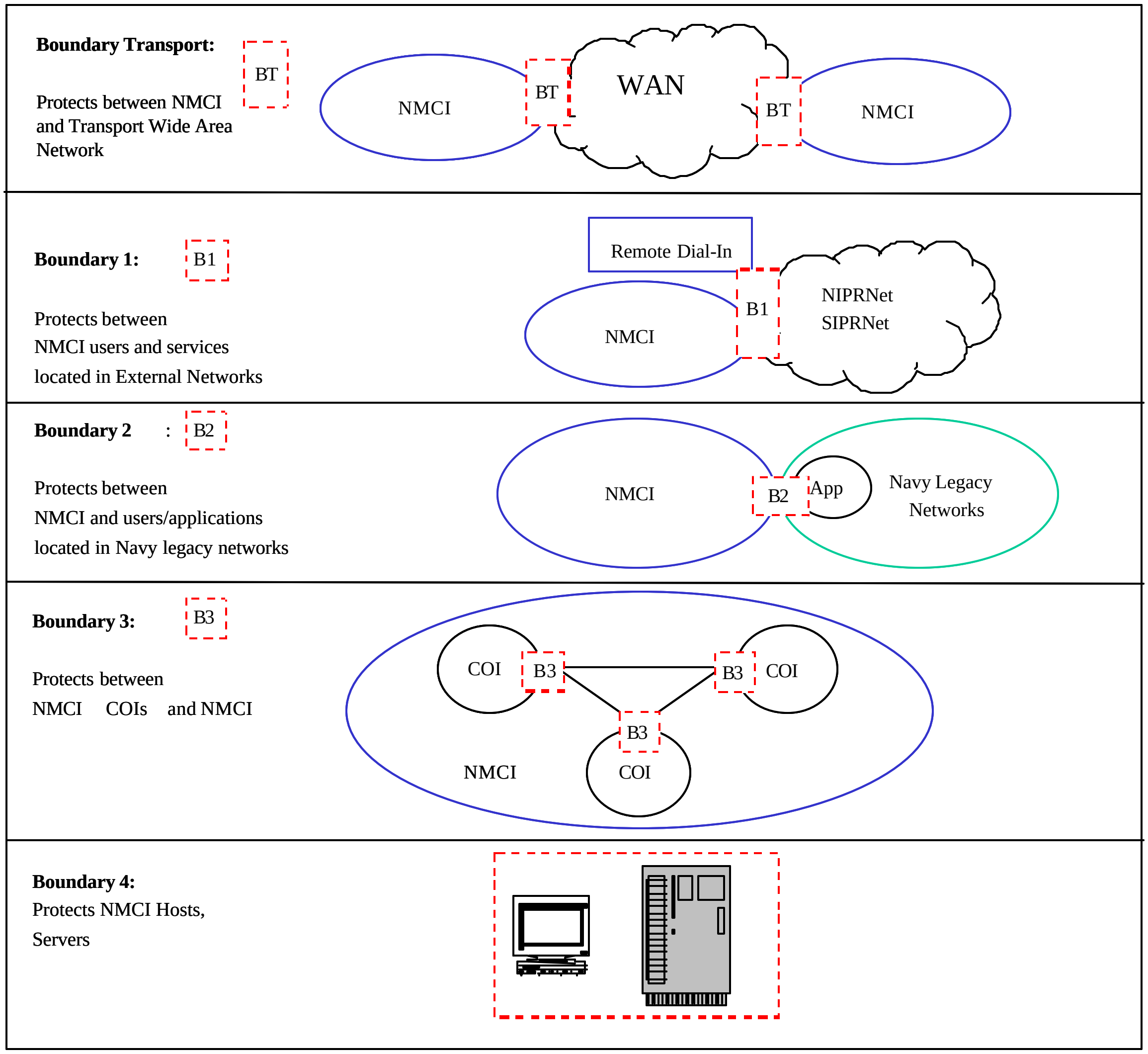
# 4.0 Interfaces

Interfaces to network infrastructure components are commonly identified by reading component specifications. Proper interfacing with the two Enterprise infrastructures of interest (NMCI and TFW Portal) is required to ensure that the infrastructures continue to operate according to their original design. This section seeks to identify infrastructure interfaces, API's, and application specifications for the various types of applications that will share the NMCI Windows 2000 desktop and Task Force Web Portal network environment. Developer responsibilities and common approaches to these interfaces will be enumerated in an effort to protect, respect and maximize our investment in the common Enterprise network infrastructure. The goal for a developer should be to develop NMCI and TFW Portal sensitive applications that will work securely and harmoniously with common network resources. An overview of the interfaces should help a developer gain the understanding needed to properly interface with Enterprise architectures, affording both developer and user a successful experience with the new Navy Enterprise. An interface compliance checklist (NMCI/ TFW Portal) can be found in Appendix 0X. Excellent resources that define these specifications are, the "Windows Logo Program" that may be found on the Microsoft's developer network website at http://msdn.microsoft.com/certification/download.asp, the Microsoft Platform SDK (Software Developer Kit) that documents the Win32 API, and Microsoft's ADSI (Active Directory Service Interface) model (see Appendix X0 for industry references to common interfaces used in NMCI/ TFW Portal).

**Desktop Application Specification (The Legacy Approach)**

Although it is the intent of this document to provide guidance towards developing web enabled applications, it will be, in some cases necessary to develop or modify existing desktop applications to ride on the NMCI infrastructure. This section describes the standard Windows 2000 API's used in NMCI workstations and discusses NMCI's use of Novadigm Radia (a software distribution system) and Active Directory technologies to manage software availability to a workstation or an end user of NMCI.

Desktop applications developed for NMCI's Windows 2000 environment must undergo an ISF certification process (enumerated in Appendix XXX) in order to be "pushed" on to workstations via a Novadigm Radia instance (see Appendix XXY for a sample install script and what is required). The NMCI network, monitored by ISF will protect connected user workstations, data, and application servers if and only if guidance is headed by developers or users interfacing with the network. Both applications and users will be controlled as objects and removed from participation in NMCI should they violate policy or specification..



**Boundary Transport:** BT

Protects between NMCI and Transport Wide Area Network

**Boundary 1:** B1

Protects between NMCI users and services located in External Networks

**Boundary 2** : B2

Protects between NMCI and users/applications located in Navy legacy networks

**Boundary 3:** B3

Protects between NMCI COIs and NMCI

**Boundary 4:**
Protects NMCI Hosts, Servers

? Desktop Apps Interface Diagram goes here 1

Standalone Application (no network connectivity while running, just install and use)
A standalone application, for the purpose of this section may be defined as an application that requires installation on an NMCI workstation but does not require use of the network for its operation. An example could be the common Windows "calculator" application on most Windows computers. This type of application does not interface with any Windows 2000 Services or network infrastructure resources/objects.

Windows 2000 Interface specification.

Win2k standard desktop specification is provided by Microsoft at
http://msdn.microsoft.com/certification/download.asp

The desktop specification clearly outlines what a developer will need to qualify for a "Certified for Windows" logo.

NMCI lockdown policy is highly restrictive to both the end user and the application and will allow writing to the disk only during the Novadigm Radia push, applications and end users are only allow to write so certain portions of drive C as follows

- NMCI Desktops are set with the NMCI ISF screen saver. This cannot be changed by the desktop user.
- Users cannot create folders sub to the root of C:
- Users can create new folders sub to C:\PROGRAM FILES; however, most existing folders under C:\PROGRAM FILES are read only.
- All operating system level files (autoexec.bat, WINNT directory, etc.) are not available for update by applications.
- Desktop users are not allowed to make changes to application files. Application files are distributed to the user's desktop using Active Directory, Novadigm Radia, and Gold Disk processes.

Applications deployed to NMCI clients should be placed in a folder below C:\PROGRAM FILES.

Applications to be used in NMCI need to be packaged with Novadigm Radia. Radia delivers applications to the PC without being affected by file permissions because it runs under the system account.

Application data should be stored in the user's My Documents folder. The location of the My Documents folder should be obtained programmatically because this will not be the same for all users – for example Terminal Services users have their My Documents folder re-directed to their home folder on the network. The location of folder is defined in the following registry key:

*HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders\Personal*

The location can also be obtained using the following Visual Basic, C/C++ function:

```
SHGetFolderPath(NULL, CSIDL_PERSONAL, NULL, 0, szPath);
```

To ensure that NMCI workstations are both secure and stable, users (and applications) are allowed to write in only designated directories on their local hard drive. These permissions are enforced using the Windows 2000 Group Policy.

For a complete list of the current directory permissions, see Appendix 6, Directory Permissions

Summary of Responsibilities
NMCI Gold Disk & standard image interfaces
Network Sensitive (requires an NMCI network connection to run)
Network related API's other than standard Win2K API's can ISF identify any more that developers need to be especially aware of to write network aware enterprise code?
Microsoft Active Directory Service Interface (ADSI) may prove useful for proper understating of the Enterprise benefits of AD and can be found here:

(http://msdn.microsoft.com/library/default.asp?url=/library/en-us/netdir/adsi/active_directory_service_interfaces_adsi.asp )

Authentication and login (permissions & control)
Group Policy

Group policy eases managing the ongoing change and configuration issues that arise as administrators try to ensure that people are productive as they use their computers to complete their day-to-day work. Group Policy allows the administrator to stipulate users' environments only once, and then rely on the operating system to enforce them thereafter. Group Policy objects are not profiles. Profiles are user environment settings and are configurable by the user. Policies are standards configured by the administrator that are applied during computer boot-up and user logon. They specify system behavior and restrict what users are allowed to do. There are local and non-local policy types. Local policies are stored locally, within the computer's registry. Non-local policies are stored in Active Directory (AD). Local policies will not be configured within the NMCI environment.

Group Policies are processed first at the site level, then the domain level, and finally at the organizational unit (OU) level. The administratively specified order determines the Group Policy settings that a user or computer actually receives. Furthermore, policy can be blocked at the Active Directory domain, or OU level.

Application of Group Policy can be filtered by the use of security groups. The location of a security group in Active Directory is irrelevant to Group Policy.

## 4.1 NMCI Group Policy Objects

### 4.1.1 Application of Group Policy Objects

Within NMCI, Group Policies will be linked to the area(s) to which they apply:

User and Computer specific Group Policies will be linked to the Command Level OUs

Domain Controller specific Group Policies will be linked to the Domain Controller OUs

General server Group Policies will be linked to the Application Services OU

Specific application server Group Policies will be linked to the appropriate OU under Application Services

Legacy Apps Group Policies will be linked to the Command Level OUs, as needed

Workstation preference Group Policies will be linked to the Command Level OUs

The following tables display the NMCI Group Policy links, by domain: (See Appendix XXX for Group Policy Specifications)

## 4.2 Group Policy Object Creation

NMCI workstation and AD object lock down will be achieved via restrictive Group Policy Objects implemented at both workstation startup and user authentication.

## 4.3 Terminal Services Group Policy Object

Terminal Services within the NMCI environment will use a Terminal Services Group Policy linked to the Application Services/Terminal Server OU. This Group Policy includes required computer and user settings for terminal services sessions. The Group Policy will be configured following the instructions included within the NMCI Terminal Services documentation and makes use of the loopback mode option.

## 4.4 Workstation Preference GPO Settings

### *P_XXXX_WKSTCompPref_v5.05 Settings*

This table lists the workstation preference Group Policy computer configuration settings. These are the only configured settings within the Group Policy. All other settings are "Not Configured" or similar.

| Section | Name of Key | Setting |
|---|---|---|
| Computer Configuration\Administrative Templates\Windows Components\ | | |
| Windows Installer\ | Disable browse dialog box for new source<br>(Check to force setting on) | Enabled |
| | Disable patching | Enabled |
| | Enable user control over installs | Disabled |
| | Enable user to browse for source while elevated | Disabled |
| | Enable user to use media source while elevated | Disabled |
| | Enable user to patch elevated products | Disabled |
| | Allow admin to install from Terminal Services session | Enabled |
| System\ | Disable Autoplay (Select CD-ROM) | Enabled |
| | Don’t display welcome screen at logon | Enabled |
| | Disable legacy run list | Disabled |

### P_XXXX_WKSTUserPref_v5.06 Settings

This table lists the workstation preference Group Policy user configuration settings. These are the only configured settings within the Group Policy. All other settings are "Not Configured" or similar.

| Section | Name of Key | Setting |
|---|---|---|
| User Configuration\Administrative Templates\ | | |
| Desktop\ | Prohibit user from changing My Documents path | Enabled |
| Control Panel\ | | |
| Add/Remove Programs | Hide the "Add a program from CD-ROM or floppy disk" option | Enabled |
| | Hide the "Add programs from Microsoft" option | Enabled |
| | Hide the "Add programs from your network option" | Enabled |
| Display | Disable changing wallpaper | Enabled |
| | Hide Screen Saver tab | Enabled |
| | Screen saver executable name(Executable name: NMCI.SCR) | Enabled |
| | Password protect the screen saver | Enabled |
| | Screen Saver timeout (Setting:  300 seconds) | Enabled |
| System\ | Do not run specified Windows applications (Add: autorun.exe, install.exe, setup.exe) | Enabled |
| Logon/Logoff | Disable legacy run list | Disabled |

Directory Permissions

NMCI Directory Permissions are defined in Appendix XXX, with the following legend:

FC = Full Control

R = Read

E = Execute

W = Write

D = Delete

The Directory Permissions List Requires ISF maintenance to remain up to date – HOW do we address this as a group?

Registry Permissions

NMCI Registry permissions are enumerated in Appendix XXX, with come with basic guidelines.

Installer agent (Novadigm Radia, run with administrative privileges) can modify the desktop registry during application installation only.

Users and applications cannot modify registry keys other that those specified in Appendix XXX

Active Directory User Objects

Any client that is locating a Windows 2000 service should query Active Directory to obtain binding information for the services that are of interest.

In Windows 2000, services publish their existence via objects in Active Directory. The objects contain binding information that applications use to connect to instances of the service. To access a service, an application does not need to know about specific computers; the objects in Active Directory include this information. An application queries Active Directory for an object representing a service (called a connection point object) and uses the binding information from the object to connect to the service.

In a distributed system, the computers are engines; the interesting entities are the services that are available. From the user's perspective, the identity of the computer that provides a particular service is not important. What is important is accessing the service itself.

To take advantage of the service-centric view afforded by the Active Directory Service, client applications must:

- Query Active Directory for accessible services.
- Present these services to the end user or automatically select the appropriate service connection point object.

- Connect to the service using the binding information contained in the selected connection point object.

For examples and more detailed information, see the section titled "Searching Active Directory" in the Active Directory Programmer's Guide at http://msdn.microsoft.com/developer/windows2000/adsi/actdirguide.asp .

OSI Model, Network connectivity / ports

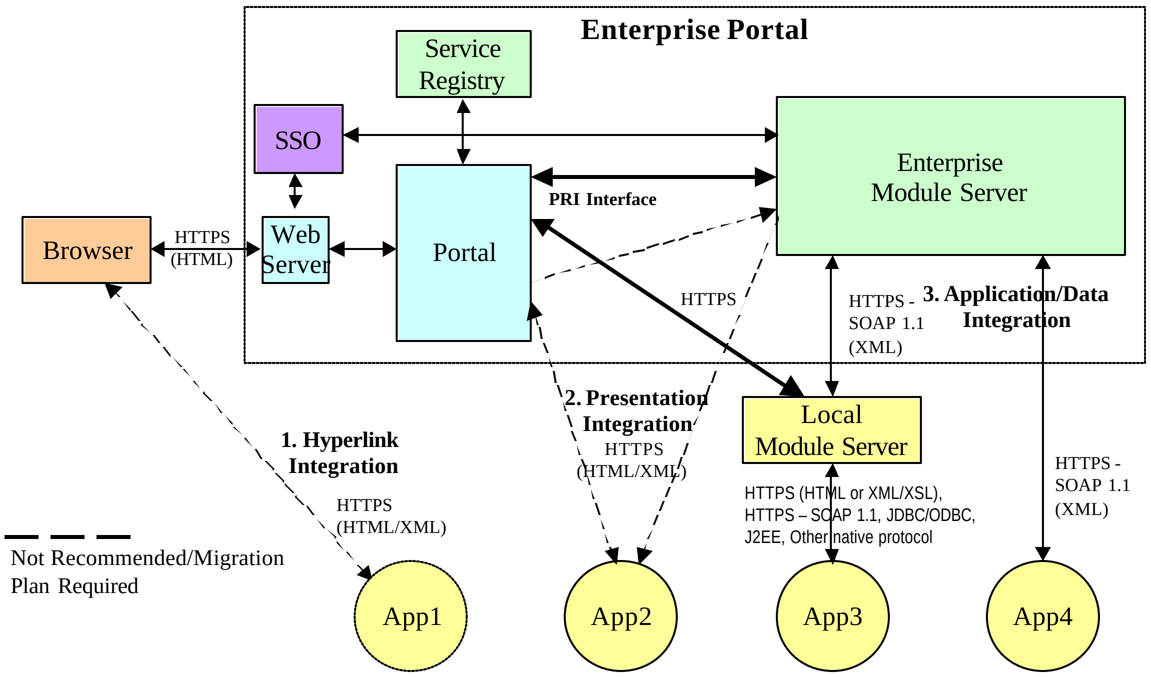
The OSI model illustrates the various network layers (An interface diagram is needed here as appropriate to NMCI architecture).

Application Interface dependencies & Portability

Summary of Developer Responsibilities

Developers are ultimately responsible for their applications running on the NMCI infrastructure. Applications may be rolled back by ISF if they impact the performance of the network, compromise security or are otherwise non-compliant (see application checklists, rules and regulations). It is highly recommended that developers ensure their applications are good citizens and follow guidelines to protect user's data.

Web (The Recommended Approach) (NEED TFW REP RECOMMEND WHAT TO KILL & KEEP HERE…)



? TFW Portal Interfaces 1

The above diagram shows the various application, portal and user interfaces for the various levels of TFW integration (1 to 3). As with NMCI interfaces, Active Directory is the common element that binds these objects together.

### Client / Server Interfaces

The required level of integration is LEVEL 3 (Application Data Integration), defined as Application/Data Integration involves a more closely coupled integration of the application with the Enterprise Portal. This integration level requires that the application move toward supporting what are commonly known as “Web Services”.

Application/Data Integration is the TFWeb-preferred level of integration. All application content is provided through Service Modules that reside in a Module Server, either the Enterprise Module Server or a Local Module Server. These types of Service Modules act as lightweight connectors, exposing some fine-grained portion of application functionality in a manner that is compliant with the Enterprise Portal. Application logic and data continue to reside within the existing application and data layers, and not within the Service Module. When accessed by a user, all application content is rendered within a pane of the portal (an IFRAME) on the user’s desktop. Access to all services is controlled by SSO, in the case of Service Modules hosted in the Enterprise Module Server. Local Module Servers are responsible for controlling user access to local Service Modules. The user is able to directly interact with the application appearing in this pane. All communication between the user and the application must flow through both the Enterprise Portal and a Module Server. All communication between the service module and the back-end application must utilize the Simple Object Access Protocol (SOAP) v1.1 XML messaging standard.

When invoked, the Service Module interacts with the backend application or web service (as described in Section 7), and formats the results of the request into the appropriate XML/XSL response. Additionally, because all content is passed through the portal, any service module or application providing XML/XSL content will be converted to HTML by the XML rendering engine that resides in the Enterprise Portal.

App4 in Figure 2-1 employs Application/Data Integration. The application exposes a SOAP interface to the Service Module, and the Service Module implements the PRI interface to the Enterprise Portal. App3 in Figure 2-1 may also be integrated at Level 3 as long as the Local Module Server implements the PRI to the Enterprise Portal, and also exposes a SOAP interface when communicating with other applications or services. In this case, both App3 and App4 would be considered "Web Services", providing interoperability capabilities fully aligned with the TFWeb vision. Level 3 integration may appear the same to the end-user as a Level 2 integrated application, as shown in Figure 2-3.

A description of the Service Modules will be registered with the global Service Registry to provide the Enterprise Portal with quick access and search capability. Section 7 provides service developers with additional details of how to build Application/Data Integration Service Modules.

Application/Data Integration is commonly referred to as Level 3 integration.

API's (common approaches)
- PRI
- SOAP
- J2EE (need guidance references)
- .NET
- XML (defacto standards, don CIO, interoperability)

### 4.4.1 Process PRI Request

The Portal Request Interface (PRI) will place the XML PRI request in the HTTP header as variable PRIDataRequest containing an XML message. The service module must determine if the PRI request is present and if it is valid. Standard Java classes are available on the Open Source Site (https://tfw-opensource.spawar.navy.mil/RegRepTeamApps/WebHelp ) to allow for this validation, see appendix B for a description of this class, PRIRequest. If the PRI request is not present in the HTTP header or the PRI request is invalid, the service module must exit with a 403 error.

All communication between the Enterprise Portal and the module server occurs using the Portal Request Interface (PRI) specification. The PRI, as illustrated in Figure 0-1, is based on open, industry standards – specifically HTTPS and XML. The portal sends an HTTPS request to the URL that corresponds to the service being called. The request is an HTTPS "post" or "get", with an additional XML message passed as an http header parameter. The XML message contains session channel context information for the request, such as the user identification and delivery channel, that the service may require in order to process the request.

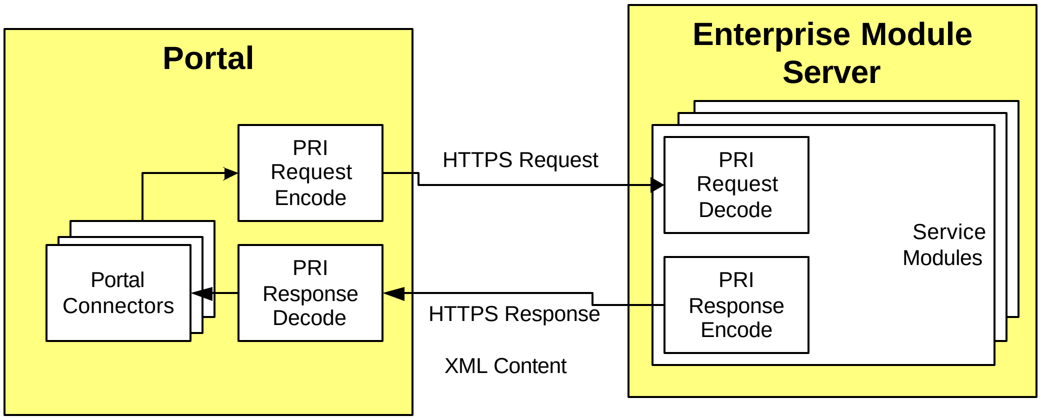


**Figure 0-1: Portal Request Interface**

After processing the request, the service sends a standard HTTPS response back to the portal. The content of this HTTPS response is either HTML, or XML and an XSL style sheet, which the portal will then render and display to the user. Included within the header of the HTTPS response is an XML message that includes some information and instructions that the portal requires in order to render the response.

The PRI interface currently does not provide the capability to dynamically set the timeout value to wait for a response from the module server for each request / reply transaction. The timeout value is currently a static value, configurable by the portal administrator

The following table provides additional detail on the PRI Request.

**Table 0-2: PRI Request Data Definition**

| Data Element Name | Size / Format | Description | Notes |
|---|---|---|---|
| Standard information to be sent as part of the HTTPS request | | | |
| Standard HTTP Request Headers | See section **Error! Reference source not found.**. | Standard HTTP headers that the portal received from the client browser. | HTTP headers are passed in the request from the source to target system reflect the header information received from the client browser via web infrastructure. |
| PRIDataRequest data elements sent as a XML message in the HTTP header | | | |
| UserID | 200 characters (Alphanumeric) | The portal user's identification based on the Navy flat name space schema. | The portal framework must determine the user ID from either the client browser or the directory service. |

| Data Element Name | Size / Format | Description | Notes |
|---|---|---|---|
| RoleAssignments | Array of Alphanumeric Strings | The user's role assignments. | The portal framework must determine the user role assignments from the directory service. |
| PortalLocation | 80 characters / Alphanumeric | The location of the portal instance. | Either "ashore" or "afloat". For the Pilot, the portal is not required to dynamically determine this value. It may be manually configured within the portal connector template instance. |
| Client | 80 characters / Alphanumeric | The content delivery channel to the client. | For the Pilot, the only supported delivery channel will be "browser". The portal is not required to dynamically determine this value. It may be manually configured within the portal connector template instance. |
| CheckBandwidth | 10 characters / Alphanumeric | A flag to inform the service module that communication bandwidth restrictions may exist for this request. | This value will be either "true" if the service module is required to verify bandwidth availability, or "false". The portal is not required to dynamically determine this value. It may be manually configured within the portal connector template instance. |
| SessionID | A 32-digit Globally Unique Identifier (GUID) in the format "nnnnnnnn-nnnn-nnnn-nnnn-nnnnnnnnnnnn". | A session identifier for the portal user's browser session. | The portal dynamically generates and maintains this value. Applications may use this to maintain state. |
| ClientStyle | 200 characters / Alphanumeric | Reference to the Portal stylesheet that corresponds to the users current template | Allows the application's page to maintain a consistent look and feel with the Portal |

### 4.4.2 Build Application SOAP Request

The service module will create the necessary XML that is required by the application SOAP server. This is completely dependant on the services that are available by the application. The XML will be encapsulated within the SOAP request.

#### 4.4.2.1 Prompt for application username and password

If the application is not integrated with the TFWeb SSO product, see section 6.3(TFWeb SSO Architecture), and the application requires users to be authenticated prior to use, the service module must display a userid/password prompt. The user/password entered will be passed to the application in the SOAP request. The service module cannot trust the userid, which is passed as part of the PRI header, is valid. The prompt for a userid/password should loop between the prompt and the SOAP request to the application. If the application fails with an authorization error the script should reprompt the user for the security info. The service module should prompt for the security info 3 times before failing with a security violation.

#### 4.4.2.2 Send SOAP Request to Application

Two types of requests can be made to the application for data.

##### 4.4.2.2.1 Enterprise Module Server Request

The service module makes the SOAP request to the back end application. By the nature of the request SOAP an XML document will be return in the response.

##### 4.4.2.2.2 Local Module Server Request

If the request is not going to use SOAP, then the request will be made from a Local Module Server, service module. The request from the Local Module Server is not required to use XML and the request/response mechanism must be defined between the service module and the back end application

#### 4.4.2.3 Application Processes Request

This is not part of the service modules process. The application that a request is made of, as described in Section 0, will have to accept the SOAP request acting as a SOAP server and process the request. The application will return a SOAP response.

#### 4.4.2.4 Receive SOAP Response

The application will encapsulate the response with the SOAP response.

### 4.4.2.5 Process Application result data

The data returned by the application may need to be reformatted or transformed to some extent by the service module prior to returning to portal. This should be kept to a minimum, as the service module should not contain any business logic. The portal will transform any XML to HTML for display in the browser.

### 4.4.2.6 Build PRI Response

The service module will create a PRIResponse object, this is optional, it is only required for application error reporting. (see appendix B for description). The PRI Response will be in the HTTP header for processing by the Portal Request Interface (PRI). The following table explains the fields in the PRIDataResponse

**Table 0-3: PRI Response Data Definition**

| Data Element Name | Size / Format | Description | Notes |
|---|---|---|---|
| Standard information to be returned as part of the HTTPS response. | | | |
| Standard HTTP/HTTPS Response Headers | See Section **Error! Reference source not found.**. | Standard HTTP response headers from the service module. | HTTP/HTTPS headers are passed in the response from the service module. |
| Standard HTTP Body Content | Level 1 & 2: XML/XSL (preferred) or HTML<br>Level 3:<br>XML/XSL | The content returned from the service to be rendered by the portal and displayed in the client browser. | The service module must respond with portal compliant HTML. Please see the TFWeb Portal Service Architecture Design Document for more details concerning portal compliant HTML/XML requirements. |
| PRIDataResponse data elements sent as a XML message in the HTTP header (Optional) | | | |
| ReturnCode (Optional) | Numeric (Integer) | A numeric value optionally returned by the service module to indicate success or failure of the operation. | The following are valid return code values:<br>0 – Success<br>1 – Informational<br>2 – Warning<br>3 – Fatal<br>Please see Section xxx for more information. |

| Data Element Name | Size / Format | Description | Notes |
|---|---|---|---|
| ReturnMessage (Optional) | Alphanumeric (String) | An alphanumeric string optionally returned by the service module that provides a textual description of any error condition that may have occurred. | |
| Timeout (Optional) | Numeric (Integer) | A numeric value optionally returned by the service module to specify, in seconds, the default request timeout value for subsequent portal to module server requests made by that specific portal connector. | A numeric integer value, greater than zero. This is a future capability that will not be supported in the Pilot. |

### 4.4.2.7 Application Results Return and Error Handling

Lastly, the service module will return the results from the application request and the PRI Response to the portal. The portal will format this data for rendering within the browser (See Section 3)

## 4.5 Error Handling

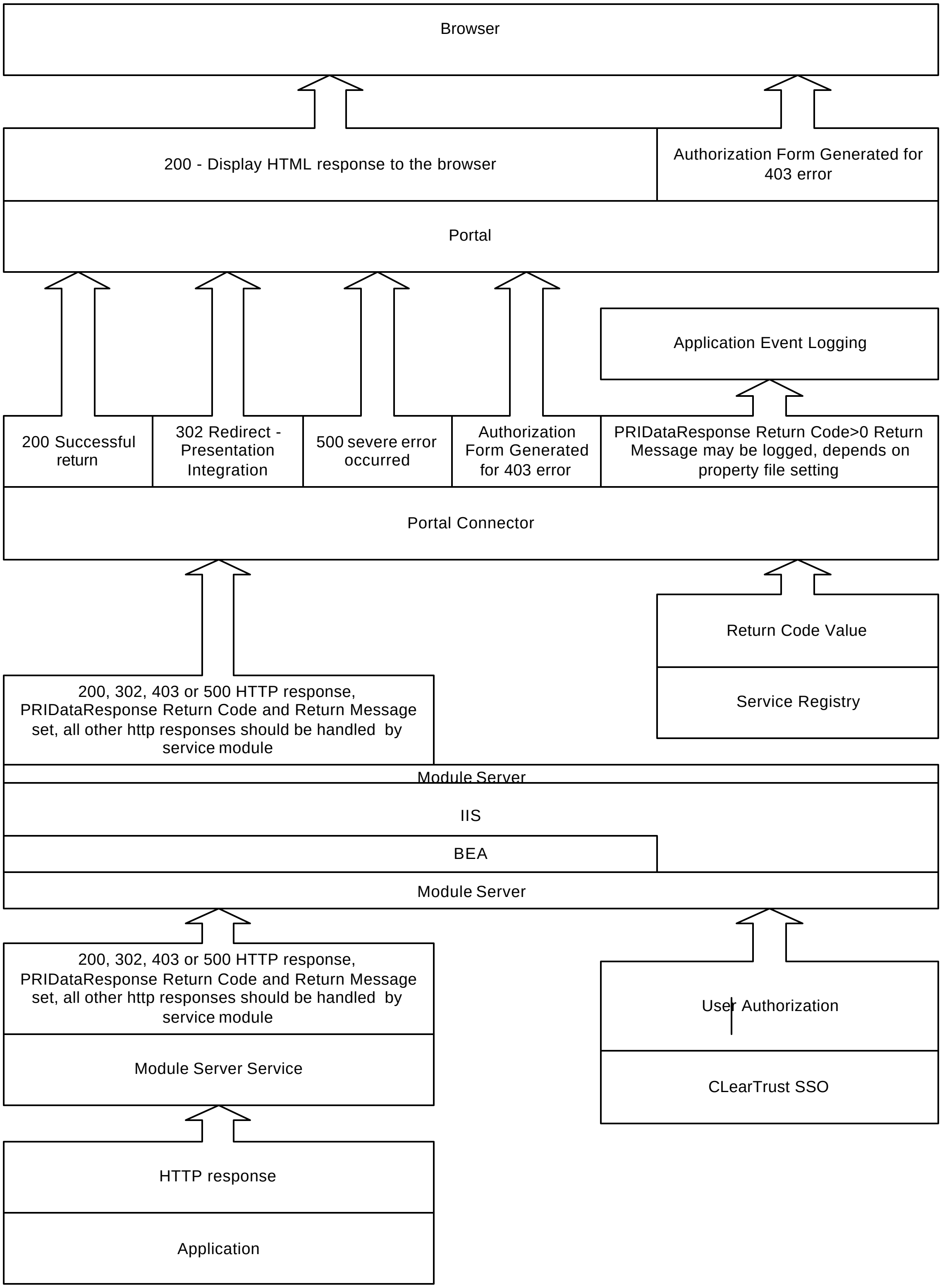

Browser
200 - Display HTML response to the browser
Authorization Form Generated for 403 error
Portal
Application Event Logging
200 Successful return
302 Redirect - Presentation Integration
500 severe error occurred
Authorization Form Generated for 403 error
PRIDataResponse Return Code>0 Return Message may be logged, depends on property file setting
Portal Connector
Return Code Value
Service Registry
200, 302, 403 or 500 HTTP response, PRIDataResponse Return Code and Return Message set, all other http responses should be handled by service module
Module Server
IIS
BEA
Module Server
200, 302, 403 or 500 HTTP response, PRIDataResponse Return Code and Return Message set, all other http responses should be handled by service module
Module Server Service
User Authorization
CLearTrust SSO
HTTP response
Application

**Figure 0-4*****: Application Return Response and Error Handling***

The above figure will be explained from the application response at the bottom of the diagram up to the browser. Each of the steps along the way has the potential to generate an error and pass the error to the calling program this explanation will follow that path.

### 4.5.1 Application

The application will generate a response based on the HTML or SOAP request. The response will be passed to the Module Server for processing. Any error can be returned to the Module Server Service Module, however, only 200, 302, 403 and 500 messages should passed up the chain from the Module Server Service Module.

### 4.5.2 Service Module

The following table will describe the Service module's response to the HTTP request from the portal. In general, the service module should trap any errors the application might send it and send informative error messages back in the PRIDataResponse (for logging purposes) and the HTTP message body (for end user viewing).

**Table 7-5: Service Module Response**

| Field | Value | What the setting indicates | How the Portal Connector interprets the field |
|---|---|---|---|
| HTTP Response Code | 200 OK | Successful communication between the Portal and the Module Server. | Utilize the PRIDataResponse and the HTTP Response Body as described below. |
| | 302 Moved Temporarily | Type 2 (Presentation) integration is in effect. | Utilize the PRIDataResponse as described below.<br>Make a new HTTP request to the redirected URL. |
| | Other | Either the Service Module encountered a serious error and crashed or IIS, BEA, or the SSO component did not allow the Service Module to execute. | Log the error.<br>Generate a meaningful message to send back to the end user. (In the case of a 403 error from the SSO component, generate an authorization request form.) |
| PRIDataResponse | Missing from response | No error occurred | Take no logging action |

| Field | Value | What the setting indicates | How the Portal Connector interprets the field |
|---|---|---|---|
| PRIDataResponse Header – `ReturnCode` | 0 or not defined | No message. No error occurred | Take no logging action. |
| | 1 | Informational message. | Log if configured to log informational messages. |
| | 2 | Warning message. | Log if configured to log warning messages. |
| | 3 | Fatal Error message. | Log. |
| PRIDataResponse Header – `ReturnMessage` | String | Message to be placed in the log. | The message that may be written to the log. |
| HTTP Response Body | HTML or XML/XSL Content. | Content displayed to the end user. | If content type is `text/xml`, parse XML for XSL stylesheet reference and use to turn into HTML. |

### 4.5.3 SSO

The SSO will be invoked by IIS plug-in to validate user access to the application. If the user is not authorized the return code from the IIS plug-in will generate a HTTP 403 response. The Module Server will not be called if the SSO does not authorize the user. The 403 responses will be passed up to the portal connector.

### 4.5.4 Module Server

The Module Server will invoke the SSO plug-in to authorize the user. If the user is not authorized to use the service the Module Server (IIS) will return a 403 response. All messages from the service are not changed by the Module Server and are passed directly to the Portal Connector.

### 4.5.5 Service Registry

The portal connector accesses the Service Registry via the service registry API to determine the information required to invoke the service module. The integer return code from the service registry API indicates one of three types of conditions:

Success (0)

Invalid Key/Service not found (10210)

Database Error (any value other than success or key not found – maps directly to the SQL Server database error code)

### 4.5.6 Portal Connector

The portal connector calls the service registry to determine the information required to invoke the service. If an error is returned from the service registry the portal connector will log the error in the application event log. Also, the error will be passed back to the portal with in the HTML response to indicate the error message. The HTTP response will be a 200.

The portal connector calls the module server to invoke the service. If an error occurs with that call the error will be logged. Also the portal connector should examine the PRIDataResponse to determine if an error occurred in the module server. If the PRIDataResponse contains an error the error should be logged in the Application Event Log.

If a 403 is returned to the portal connector by the module server the portal connector will generate an HTML response that contains a form for submission to the application owner for to allow access to the application for the user.

### 4.5.7 Application Event Log

The portal connector will log all errors to portal's logging mechanism

### 4.5.8 Portal

The portal will display the HTML to the user. The error returned from the Portal Connector will be displayed within the pane for the application. If the portal connector returns XML/XSL the portal will process the XML/XSL into HTML for display to the user.

### 4.5.9 Browser

Users browser will render all HTML returned by the portal.

## 4.6 Messaging Protocols

A messaging transport is the technology that facilitates peer-to-peer application communication using open standards. The module server will communicate with applications via the following protocols:

**Table 7-11: Module Server protocols**

| Protocol | Enterprise Module Server to application | Local Module Server to application |
|---|---|---|
| HTTPS | | Optional |
| SOAP | Required | Optional |
| Other RPC | | Optional |

Because the communication between the application and the module server must take place over HTTPS an HTTPS server is required on the application server. This can be any server that will allow communication over HTTPS.

### 4.6.1 SOAP

In the scope of the Task Force Web Portal, the term ‘Web Service’ is used to describe a piece of application functionality that is exposed to the Portal environment. To implement a functional, Level-3 integrated Web Service, the Service Developer will need to understand the role of the Simple Object Access Protocol (SOAP) specification. SOAP 1.1 is a lightweight protocol for exchanging structured and typed information between peers in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined data types, and a convention for calling and receiving responses from a SOAP server. From the perspective of the Service Developer, the role of SOAP is to serve as the messaging framework between the EMS Service Module and the backend Application entry point. All Level-3 integrated Web Services should execute in a synchronous manner for the pilot. You can think of SOAP as a standard way of packaging up the method calls and their corresponding return values.

### 4.6.2 The SOAP Client

For Level-3 Integration, the EMS Service Module component will represent a SOAP Client. This component will be physically deployed on the Enterprise or Local Module Server and will serve as the broker between the Portal Connector and the back end Application entry point. It is the responsibility of the Service Module component to:

Decompose the PRI Request posted by the Portal Connector

Build a SOAP Request Message (A packaged Remote Procedure Call)

Send the Request Message to the back end Application entry point via HTTPS

Receive the SOAP Response (A packaged Result set or Fault)

If the call executed without error, bubble up the returned XML/XSL back to the Portal server. If a SOAP Fault is returned by the backend Application or an HTTP Error occurs, bubble up the error back to the Portal server in the PRI Response structure.

### 4.6.3 The SOAP Server

The backend Application entry point component is a SOAP Server; therefore, it is the responsibility of the SOAP Server component to:

Listen and receive SOAP Requests

Decompose the SOAP Requests and make the proper handoff (i.e. method call) to a locally implemented business logic component

Format the XML/XSL Data to be returned

Build and return a SOAP Response Message, including the XML/XSL Data as a return parameter.

### 4.6.4 SOAP Programming Interfaces

The Enterprise Module Server will provide SOAP Programming Interfaces for both the BEA Web Logic and IIS Module Servers. The SOAP Programming Interfaces provide programmatic abstractions that allow developers to build and manage SOAP Messages without having to work directly with the XML Document Object Model. The TFWeb Portal will only support SOAP 1.1 standards. The BEA Web Logic J2EE Application Server provides built in SOAP Services that can be utilized by JSP or Servlet based Service Modules. Example Service Module source code using the BEA Web Logic SOAP Services can be downloaded from the Registry Module Server Developer's Network on the BEA Level 3 Integration Examples and SOAP Home pages. The Microsoft SOAP Toolkit 2.0 will provide the SOAP Programming Interfaces for ASP-based Service Modules deployed in the IIS Module Server. Example Service Module source code using the SOAP Toolkit 2.0's interfaces can also be downloaded from the Registry Module Server Developer's Network on the IIS Level 3 Integration Examples and SOAP Home pages. The Registry Module Server Developer's Network is located at:

https://tfw-opensource.spawar.navy.mil/RegRepTeamApps/WebHelp/

### 4.6.5 Service Registry

The Task Force Web Service Registry is the physical directory that stores and manages information and metadata about Web Services. The three major data entities that are managed by the Service Registry are

Echelons - The Echelon is the top-level entity and logically maps to the major organizational divisions within the Navy. Contact, Address and general "white page" information is stored at the Echelon level. Each Echelon can own 0-to-many Application entities.

Applications - The Application entity falls under the Echelon entity and logically maps to a software application managed by the Echelon. Each Application can own 0-to-many Web Services.

Web Services – The Web Service entity falls under the Application and logically maps to a specific piece of functionality provided by the Application. The Service Registry provides extendable storage facilities for Web Service technical metadata through the use of Technical Specifications. Technical Specifications can be registered and instances attached to a Web Service.

## 4.7 Session Management

The Module Server does not provide persistent storage space to service modules to maintain state between user connections. This is primarily due to the limitations imposed by the clustered and dynamic load balanced configuration of the Enterprise Module Servers.

# 5.0 Legacy System Migration and Evolution

This section focuses on efforts required to migrate existing Navy Legacy Systems into NMCI and TFW in the near term. It also addresses the long term Navy Enterprise Domain IT goal~~Goal~~: Navy Systems that are Enterprise Domain Information Technology Solutions, providing web services and employing XML/SOAP interfaces within and between Enterprise Domains.

Sections 5.1 through 5.5 describe process activities associated with integrating Navy Legacy Systems with TFW and NMCI, including Legacy Rationalization. Section 5.6 addresses technical steps to XML and SOAP enable the Legacy Systems. Section 5.7 addresses steps to meet the above goal. The Section 5.7 steps address migration and retirement of legacy systems into a next generation Navy Enterprise Domain System.

## ~~Near Term Solution to advance toward the Goal: XML/SOAP-Enabled Navy Legacy Systems~~

~~Most Navy legacy systems require modification of interfaces to make them meet TFW and NMCI requirements. By XML/SOAP-enabling these systems, using DoD standard XML, these systems may be able to interoperate on the TFW portal in an NMCI environment. This section describes steps required to XML/SOAP-enable Navy Legacy Systems.~~

~~XML/SOAP-enabling steps~~

~~[MOVED FROM ARCHITECTURE SECTION]~~

~~Web Services from legacy applications~~

~~? Identify key functionality that is required to support existing and future users of the application.~~

~~? Decompose the application into discrete services that provide the functionality identified above.~~

~~? For each service, search the enterprise registry to determine if similar service already exists and is suitable.~~

~~? For each service, define the SOAP/XML request/response messages.~~

~~? Implement a module that accepts the SOAP request message, performs the indicated service and generates the SOAP response message.~~

~~? Test the service~~

~~? Submit the module to the appropriate registration authority for publishing in the enterprise service registry. (Registration authority will conduct testing and verification prior to publication).~~

~~[END]~~

~~Some Navy legacy systems may require additional system architectural changes to comply with NMCI and TFW requirements. The next section describes those architectural requirements.~~

~~NMCI and TFW system architectural requirements~~

## ~~5.1~~5.1 Applications and Databases of concern to NMCI and TFWeb

The first step in dealing with a legacy application is to determine if it falls under the cognizance of NMCI, TFWeb or both.

NMCI is concerned with all the applications and databases on the NMCI network, with the exception of those on science and technology workstations. However, there will be networks at the Navy that are not under NMCI control. (True??)

TFWeb is concerned with applications and databases that are candidates for web or portal enablement. In Section 5.2.4, Web Enablement Determination, there are criteria presented for Web enablement. If an application does not meet the criteria in this section, then it is not an application of concern to TFWeb.

The NMCI and TFWeb criteria result in a large number of applications and databases that will be of concern to both NMCI and TFWeb. However, there will applications and databases that are of concern to one but not the other. Therefore, this section will note when the guidance comes from NMCI and when it comes from TFWeb. Applications and databases that are of concern to both NMCI and TFWeb should consider all of the guidance provided in this section. Applications and databases that are only NMCI or only TFWeb should consider the appropriate subset of this guidance.

## ~~5.1~~5.2 Application Owner/Analyst Guidance

The process for migrating an existing application into the Navy portal is designed to ensure that the target application meets all portal standards, security requirements, does not utilize a data environment duplicative of an existing authoritative data source, and does not provide a duplicative service.

The process begins with the service provider determining the applicability of migrating the application to the Navy portal. Next, a review of existing services and data sources is conducted to identify various duplication issues. Once the decision is made to migrate the application to the Navy portal, the developer submits an Intent to Migrate notification to the TFWeb Application Migration Customer Support (AMCS) team. An AMCS officer will be assigned to the application who will assess the application, help to identify overlapping applications and data sources, and assist in compiling the Request to Migrate for submission to the Task Force Web AMCS team. After review, the AMCS team will forward applicable portions to the beta test labs for final technical review prior to integration. This chapter will discuss the AMCS processes and the specific steps required to complete each.

## ~~5.2~~5.3 Pre-Service Registration Phase

Before starting the TFWeb integration process, the developer must answer for themselves a number of questions:

- What is my TFWeb integration goal (see ~~this section~~ Determining TFWeb Integration Goals )?
- What is my virtual interest group (see Determining Communities of Interest ~~this section~~)?
- Is there an approved DoN/DoD application or service already in existence that provides this service/content (see Reviewing Existing Services ~~this section~~)?
- How do I find other registered services on the Enterprise Portal (see Market Review of Existing Services and Content ~~this section~~)?
- How is "best of breed" determined (see Registered Services and "Best of Breed" Determination ~~this section~~)?
- Is my application/service already web-enabled (see Section Web Enablement Determination ~~this section~~)?
- If so, now what (see Existing Web-Enabled Applications ~~this section~~)?
- If not, should I web-enable it (see Non-Web-Enabled Applications ~~this section~~)?

Once these factors have been determined, the program, application, or content manager will have the data needed to determine what migration plan/POA&M will be required to achieve their targeted level of integration.

### ~~5.2.1~~5.3.1 Determining TFWeb Integration Goals

What is a "web-enabled" application? This term is often misunderstood. A "web-enabled" application is simply an application or service that is accessed within the context of a browser and is based on Internet communications standards. This includes, but is not limited to, applications/technologies such as Java (beans, applications, scripts, applets (signed), server pages), Active Server Pages (ASPs), ActiveX components (signed), multimedia, and other approved plug-ins.

Regardless of your current web posture, there are certain key things that a developer, program, application, or content manager must determine before considering integration into the Navy Enterprise Portal:

- Defining Communities of Interest.
- Market review of existing services and content.
- Supportability and Maintainability.

### ~~5.2.1.1~~5.3.1.1 Determining Communities of Interest

A key objective of web-enablement is the cross-pollination of data within and across communities of interest. WEN service providers are required to determine the virtual interest group (as shown by the taxonomy in Section 5) for their application. This determination is based upon the types of information services and/or data that are common to the community's processes or business operations and whether they would benefit from web-enabling as well as portal integration and dissemination. Combining services within virtual interest groups will illustrate size, priority, and complexity of data/information and application sharing and aid in determining cost-benefit and other intangible benefits (e.g., reduction in system operator/administrator task complexity). The managers of each community of interest (e.g., ASNRDA-CHENG/OPNAV for Battleforce information requirements) will provide the developer with the location of their authoritative data source(s) through the WEN IT Governance Board/TFWeb process. The goal is to provide an integrated data environment that will use smart data replication to allow enterprise access to authoritative data sources observing the demands of limited operational bandwidth and connectivity. This environment will promote application re-use/consolidation around the authoritative data sources.

The Functional Data Manager, established under SECNAVINST 5000.36, should be an important part of this community and should be included in all application and database decisions.

### ~~5.2.2~~5.3.2 Reviewing Existing Services

### ~~5.2.2.1~~5.3.2.1 Market Review of Existing Services and Content

Program, application, service, or content managers should review existing applications (commercial or otherwise) for overlapping capabilities. Build/Buy/Re-engineering decisions should be predicated on examining the list of existing services and content and their respective descriptions to ascertain whether an existing service or content can be reutilized. In short, is there an approved DoN/DoD application or service already in existence that provides this service/content?

#### ~~5.2.2.2~~5.3.2.2 Registered Services and “Best of Breed” Determination

Information regarding current registered services and content is found at the site of the master registry or from AMCS. If a new service or content is being proposed for addition to the Enterprise Portal environment, an Intent to Migrate package (Section 8.2) must be submitted to the AMCS for review. AMCS verifies that there are no applications in the WEN environment that provide overlapping functionality or content, and that the implemented technologies, styling, and supportability requirements provide for TFWeb integration.

In the event that there is overlapping functionality, AMCS works with the application owners to understand and document the overlap and develop a migration plan. If a migration plan cannot be agreed upon by AMCS and the concerned application owners, the documentation is given with a recommendation to the TFWeb Executive Steering Group (ESG), which then determines which applications will be allowed to integrate with the portal. The ESG may also make recommendations to OPNAV and Echelon II commands to resolve application/data overlap. This decision is based upon the following criteria:

- Technical/Architectural analysis performed by an independent TFWeb engineering team. This analysis includes all relevant engineering requirements (e.g., security) as defined by this and other DoD/DoN/TFWeb guidance.
- Operational Advisory Group Analysis. An OAG comprised of members from the appropriate service elements evaluates the applications for use in their environments to meet their operational needs. Several functional groups already exist and these are utilized when possible.
- Business Case analysis. Each application provider is required to build a business case analysis for evaluation. This includes review of the funding requirements, ILS Plan, and other similar documentation.

### ~~5.2.3~~5.3.3 Supportability and Maintainability

A product that is successfully integrated into the Enterprise Portal environment will be unsuccessful if it is not adequately supported. Each WEN service provider is required to show an ILS Plan that demonstrates maintainability and supportability of their application or service.

### ~~5.2.4~~5.3.4 Web Enablement Determination

Being “web-enabled” does not mean, “TFWeb-ready”. “TFWeb-ready” connotes that the developer has followed the registration, development, integration/testing, and deployment processes laid forth in this document, and been approved by the WEN IT Governance board.

### ~~5.2.4.1~~5.3.4.1 Existing Web-Enabled Applications

Even though web enabling is not equivalent to being TFWeb-ready, being web-enabled will help accelerate the process. The main issues that will impact the developer with regards to TFWeb integration will be:

~~?~~ Implementation of web technologies (and appropriate versions) specified in the WEN Technology Baseline (e.g., Java/J2EE, Perl, CGI)

~~?~~ Presentation styling. The developer's web presence may be in conflict with TFWeb promulgated styling conventions or incompatible with the portal interface.

- Implementation of naming conventions and data interoperability standards (e.g., XML).

The ultimate decision to undertake realignment or retrofit of existing web-enabled applications into the Enterprise Portal environment is left to the program, application, or content manager. It is strongly recommended that the entire TFWeb registration process be reviewed prior to these undertakings.

### ~~5.2.4.2~~5.3.4.2 Non-Web-Enabled Applications

Before launching into an intensive integration effort to "web-enable" an existing Navy service or application, it must first be determined whether there is value in doing so. It may not make sense to web enable every application or service. In many cases, the application may not be integrated into a web-based environment, but the data it provides may be hosted on the portal as relevant content. This section of the guidance document identifies a set of criteria that can be used to evaluate an application or service for "whole" or "partial" web enabling. The following should be used as general guidance for the program, application, or content manager to determine whether or not they should endeavor to web-enable their application, and then integrate it into the Enterprise Portal environment.

The criteria identified to date include the following:

- Information Services
- Real-Time Versus Non-Real-Time
- Service/Application User environment.
- User/Administrator

It is important to note that while all applications may not require web-enabling, and therefore, do not require integration into the Enterprise Portal environment, all applications will be subject to review by the appropriate program, application, or content manager.

#### ~~5.2.4.2.1~~5.3.4.2.1 Information Services

If your application provides some content that would be usable by other elements of the Enterprise Portal, then it is a candidate for some level of TFWeb integration. It is important to remind developers that TFWeb 'web-enabling' is not equivalent to simple "web enabling." It does not necessarily mean that the application runs within the context of a browser. It may simply mean that the application offers up its data for browser-presentation rendering by the Enterprise Portal engine. It also means providing of content to a shared/common data environment.

Determination of content relevance across the enterprise is determined, in part, by identifying virtual interest group, and coordinating with the appropriate authoritative data source.

#### ~~5.2.4.2.2~~5.4.3.2.2 Real-Time Versus Non-Real-Time

The Web or Internet is not a real-time medium. There is no intention of firing a weapon from a web browser. Real-time, rapid response systems are not good candidates for web enabling. However, there may be status information from a real-time system that can and should be web enabled and made available.

If the application that you are working on does not meet the criteria for TFWeb enabling because it is real-time or near real-time, then distributed component architectures, such as CORBA, DCOM or J2EE may be appropriate. If it is used within a real-time simulation than the High Level Architecture (HLA) could be used.

#### ~~5.2.4.2.3~~5.4.3.2.3 Service/Application User Environment

Web applications are by definition multi-tiered network services that deliver content (e.g., application components or data) based on an established network, data persistence, and security model. There are application and service environments that are fundamental to operational requirements (e.g., small community of interest users that are distributed across large areas) regardless of user community size. For example, there's a community of senior flag officers that are extremely essential to operational requirements as a community of interest. These users require specific applications/data, unique to their environment, with high levels of security that must operate in a distributed manner.

#### ~~5.2.4.2.4~~5.4.3.2.4 User/Administrator

Much of the development effort of any application goes into the management interface. While required, this interface may be used by a small fraction of the total number of users. It is recommended that application owners focus first on the end-user interface to deliver as much capability to the end user as resources and time permits. Rewriting existing management interfaces often has a cost higher than any benefit gained by the managers. New applications, however, should expect that all functionality is web-based when originally developed.

## ~~5.3~~5.5 Intent to Migrate

Once the decision is made to migrate an application to the Navy Enterprise Portal, the developer must notify the TFWeb team of the intent to migrate an application or service. Completion of the~~se~~ steps ~~(Section 8.3)~~of this section is required for application-specific AMCS/AMTS support. This serves to notify all concerned of integrator or sponsor's intent to provide a given service via the TFWeb portal and allows migration tracking and preparation for receipt of required information for migration. It also helps to prioritize and focus technical support assets based on the impact of the application, timeframe of migration, and difficulty of transition. The following actions are taken by the AMCS contact assigned based on the application owner submission.

### ~~5.3.1~~5.5.1 Submission to the application information database.

This database is maintained by AMCS. AMCS will review the submission to determine if fields are complete and understandable. Descriptions should be useful and thorough. Yes/no answers may need further comment. Other information may be required to be tracked for the application. Implementation dates should be reasonably achievable. The AMCS contact for the application ensures the submission is properly reviewed and entered in the database and notifies the Echelon II contact of any changes made during the review process to their submission.

### ~~5.3.2~~5.5.2 Integration Level Appropriateness

Level 1 applications require specific detailed explanations why they cannot be Category 2 and require AMCS OIC's approval of a waiver for integration into the portal. Category 2 applications may be approved by the TFWeb Echelon II liaison for preexisting applications and for applications that require immediate rollout beyond the portal user base.

### ~~5.3.3~~5.5.3 Identify if the program uses Java, JavaScript, ActiveX, or plugins

Additional review/analysis of these issues are coordinated through the TFWeb Echelon II liaison with the appropriate AMCS department head covering technical issues. In addition, non-mobile ActiveX or plug-ins must have a satisfactory distribution plan in compliance with applicable NMCI and IT-21 policies. Determine current status of application compliance with Navy Mobile Code Policy and document any waivers currently granted.

### ~~5.3.4~~5.5.4 Examine the application database for similar programs that are currently under development

The AMCS contact reviews data sources for possible data overlap and examines overlapping applications reported by other application owners. If possible overlap exists, they interface with program managers of all concerned programs and data sources to determine exact functionality, user base, and IT requirements. If consolidation possibilities exist, they brief AMCS OIC on overlap and application/data owners' intentions to determine any further action warranted.

### ~~5.3.5~~ 5.5.5 Determine current security model and whether IATO/ATO exists, or is required

Determine what changes, if any, are required to the current application security model in order to integrate application. Is the current security model compatible with TFWeb security model (issues like SSN)? Data-only applications (using XML/XSL or HTML data pages) do not normally require an IATO/ATO. Applications utilizing mobile code always require an IATO/ATO unless that mobile code is covered by the IATO/ATO of an application previously integrated into the portal. If no changes are required, the AMCS contact ensures a copy of existing IATO/ATO cover sheet is sent to AMCS IA.

### ~~5.3.6~~ 5.5.6 Determine XML integration requirements

Evaluate the plan for design and registration of the schema and other XML documentation. Does this need to be coordinated with other commands using similar data? Ensure that the application owner is familiar with the DoN XML instruction.

## ~~5.4 Service Registration~~

~~The Service Registration package (Section 8.3) is submitted to the AMCS contact after development has been completed. AMCS performs the following items as part of the package review. This section also applies to changes required as part of the beta testing procedures prior to restarting testing.~~

### ~~5.4.1Verify completeness and accuracy of portal metadata~~

~~This should include the directory entry text, category, description, application owner, and application "customer service" contact. This is information available to any user of the portal. Is it sufficient to determine whether access to an application is required and how to obtain access? Does it address intended user base and purpose of the program? Similar programs directed at other user bases should be mentioned in the description.~~

### ~~5.4.2Verify migration plan for level of integration is submitted~~

~~Migration plan is required for applications migrating at Level I or Level II. For the pilot program, the application may enter beta testing prior to submission of a migration plan. However, the TF Web Governance Board will not approve integration into the production portal without a migration plan. A migration may be as brief or as detailed as desired; however, timeframe, critical path, and issues to be resolved must be included. Retain copy of migration plan in AMCS Echelon II notes for future reference. Brief the migration plan to AMCS OIC for approval.~~

### ~~5.4.3Ensure IATO/ATO has been updated if security model changed for TFWeb migration~~

~~Provide copy of IATO/ATO cover letter to AMCS IA for reference documentation. Submission of full accreditation paperwork is not required unless determined necessary by AMCS IA.~~

### ~~5.4.4Verify initial access control list submitted along with information describing method of updating ACL~~

~~Verify method is compatible with current portal capabilities and user expectations. If access cannot be given in a timely manner, ensure it is indicated in the application description visible to the user. Review and approve appropriate roles for application visibility. For most applications that control security at the application level, the ACL should be “All Portal Users”.~~

### ~~5.4.5Portal Compliance Testing~~

~~The AMCS liaison shall be provided a temporary login with access to key features of the application. In the event access to key areas cannot be provided due to security/access issues, alternate methods will be coordinated between AMCS and the application developer. Spot check to ensure claimed capabilities of user description are provided and significant limitations are documented. Spot-check HTML used is "portal compliant" (no frames). Record all concerns, discuss with developer or program manager. Submit any unresolved discrepancies to AMTS (for pre-beta review) or as part of the AMCS beta testing notes. This is not intended to be a thorough review of the program. Rather, it serves to ensure any obvious issues are recognized and documented prior to the beta testing process to help expedite testing~~

### ~~5.4.6Review summary of testing accomplished~~

~~Summary should include duration, type of users, type of test scripts performed, type of data used, and environment in relation to the production platform. What is the risk that the program will fail beta testing? Has there been sufficient operator testing to ensure utility in the production environment?~~

### ~~5.4.7Review portal integration information submitted~~

~~To ensure effective use of testing time and to allow maximum preparation time for testing, all required portal integration information should be submitted as part of the request. While further changes may be necessary or desirable, this allows for a package of all required information to be submitted from AMCS to the beta test site. Ensure integration module code has been provided.~~

~~Identify if substantive revisions have been made to sample code. Module code should be fully documented and readable. Evaluate code for posting to open source site (based on differences from baseline code). If review of code is required, submit request to AMTS. Are there any reusable components that should be separately maintained?~~

### ~~5.4.8DoN XML guideline compliance~~

~~If the application does not meet DoN XML guidelines, ensure migration plan is submitted and approved by AMCS OIC.~~

### ~~5.4.9Set next review date~~

~~General guideline is 1 year if all requirements met or halfway to next integration level (3 month minimum) if a migration plan has been submitted for non-Level 3 integration or noncompliant XML are used. If IATO has been submitted, review date should be prior to its expiration. By this review date, a member of AMCS will review documentation and implementation history, and determine if any additional information is required. Milestones in migration plans or further functionality development will be reviewed. Also, database information will be verified.~~

### ~~5.4.10Verify database entry is complete and accurate in AMCS application database~~

~~AMCS database information is in section 11.2.1.~~

### ~~5.4.11Technical Review~~

~~AMCS may request a technical review at any time from the AMTS or alternate source. This technical review may evaluate code base, technology, mobile code, or security among possible areas. In some cases this is used to evaluate leading-edge technology and possible unforeseen impacts on TFWeb environment. In other cases, it is used to check for compliance with TFWeb architecture in a more thorough fashion than is possible in the beta testing environment. If necessary to evaluate an application's readiness for migration, this review is completed and any discrepancies resolved prior to permission for beta testing. The AMCS is the final arbiter of whom discrepancies are required to be resolved prior to beta testing, though review of an AMCS decision may be requested from the TFWeb Executive Steering Group.~~

### ~~5.4.12Configuration Verification~~

~~Verify configuration of any local application servers or local remote module servers are documented. The ability of local infrastructure to support numbers of users intended should be documented as well as ability to scale to additional users. Any known scalability issues should be documented.~~

### ~~5.4.13Ensure application is logged in the DON CIO Data Management and Interoperability Repository.~~

~~DMIR is currently in the beta testing stage. This is an optional requirement until full functionality in the second quarter of 2002.~~

### ~~5.4.14Verify all application data structures and data interfaces are documented.~~

~~Databases should be accessible independently of application if underlying database engine and security supports. Data interfaces should also be accessible independently.~~

### ~~5.4.15Verify AMCS OIC has approved migration plan for application/data overlap.~~

~~Migration plan should address duplicative applications and data sources and their planned resolution. Migration plan is not due until final review of application after beta testing.~~

### ~~5.4.16Documentation of Developer Requirements.~~

~~Ensure developer requirements for future capability upgrades of the WEN architecture and implementation of the architecture are documented. This should consist solely of architecture or cross-application services, not those useful only to a single application. This helps prioritize additional requirements based on the ability of the developer community to capitalize upon the new features. This should only include functionality developers are currently able to utilize.~~

## ~~5.5Application/Service Delivery Phase~~

### ~~5.5.1Application Acceptance~~

~~Only applications that have completed the migration request process with TFWeb are submitted (e.g., application components, links/icons, datafill, DTDs/Schemas) for integration in Enterprise Portal.  All applications need to ensure compliance with required DoN/DoD policies in addition to those required by TFWeb. The TFWeb process does *not* supercede individual program, application, or content manager processes.  It is expected that the Enterprise Portal will receive applications that have gone through internal system engineering and logistics processes (e.g., CCB, internal testing, CM).~~

### ~~5.5.2Application Delivery~~

~~Each application is expected to deliver system and administration documentation that conforms to Enterprise Portal documentation guidelines (e.g. XML or HTML).  This includes software operation and concise loading instructions to enable users/administrators to load and administer the applications with minimum intervention. The instructions should also include load verification and load back-out procedures.~~

~~Once the application is successfully loaded into the TFWeb developmental portal environment, the developer will then continue with the remainder of the self-certification procedure, moving into the performance criteria.~~

### ~~5.5.3Application Integration~~

~~The developer is encouraged to notify TFWeb of an impending application release no later than 30 days prior to portal integration.  This gives the TFWeb team time to arbitrate schedule conflicts with other application developers.~~

~~The application integration process differs depending upon the type of application to be integrated, and the level of integration the application is achieving.  In all cases the goal is to provide the developer a process and supporting infrastructure by which they can develop, test, and certify their application(s) for use in the Enterprise Portal environment with a minimum involvement by a core TFWeb team or other external agencies.~~

## ~~5.6~~5.6 XML/SOAP-Enabling Navy Legacy Systems~~Legacy Rationalization and Levels of Integration~~

This section focuses on implementing the near term solution, XML/SOAP-Enabling Navy Legacy Systems, to advance toward the long term goal: Navy Systems that are Enterprise Domain Information Technology Solutions, providing web services and employing XML/SOAP interfaces within and between Enterprise Domains. <To reach the goal, legacy systems may need to be migrated and retired to minimize duplicate functionality and data within a domain ..replaced by a next generation system providing web services.>

TFW and NMCI requirements that Navy legacy systems must meet are stated in section 2.4. By adhering to the stated requirements, including use of DoD/DON standard XML, Navy Legacy systems may be able to support Level 3 integration on the TFW portal and be certified in an NMCI environment. This section describes steps required to XML/SOAP-enable the Navy Legacy Systems that have approval to proceed per the process steps described in sections 5.1 and 5.2. The guidance of this section must be observed in concert with process steps in section 5.3.

### 5.6.1 Architecture Considerations

Most Navy legacy systems require additional system architectural changes to comply with NMCI and TFW requirements. Section 2.0 provides some guidance in the absence of a DoN Enterprise Architecture or guidance from an Enterprise Domain Architect.

There are three basic types of Navy Legacy systems that must be modified to comply with NMCI and TFW: Mainframe, Client-Server and Web-Architected. Approaches for each type architecture are described below. Basic XML steps to be followed, which are common to all architectures, are described in the section titled, XML Procedures.

#### 5.6.1.1 Mainframe systems

Mainframe systems can be XML/SOAP-enabled fairly quickly using fifth-generation screen-scrape component solutions, such as JACADA, which recently entered a partnership with SeeBeyond. Using this type of technology, green screen interfaces can be implemented using XML and SOAP to support a level 3 integration with the TFW portal. However, NMCI requirements, which may entail a rehosting activity, cannot be met with this approach. For this situation, refer to section titled, Migrating and Retiring Navy Legacy System(s), Replacing them with Navy Enterprise Domain Solutions, for what may be the most cost-effective solution.

#### 5.6.1.2 Client-Server

Client-Server systems may require additional software components for XML/SOAP enabling, including an application server and web server. Non-web enabled applications should first strive to invest in the effort to become compliant with a Distributed Object Computing technology to select an application server. <Refer to section 2.0 for a discussion on technology selection.> However, if the application is written in a computer language or system not supported by a Distributed Object Computing technology, then they should next strive to be XML/SOAP Enterprise Architecture compliant directly. Following one of these two approaches should provide the lowest life-cycle costs for these types of applications.

#### 5.6.1.3 Web -Architected Applications

Web -Architected Applications may have the components necessary to be TFW and NMCI certifiable. A developer may simply need to follow the below steps to XML/SOAP-enable the application.

### ~~5.6.1~~5.6.2 XML Procedures

#### 5.6.2.1 XML Guidance

The following XML Guidance is applicable to support XML Design and Registry Use.

DoD COE Data Emporium and XML Registry, URL:
http://diides.ncr.disa.mil/shade/index.cfm
DON CIO Interim XML Policy, URL: TBP
DON CIO XML Guidance, URL: TBP

#### 5.6.2.2 XML DTDs (Data Type Definitions)

Applications that support XML using the older XML DTD (Data Type Definition) description of their file format first need to migrate to XML Schema. Once migrated to XML Schema, they should follow the guidelines provided under the XML Schema guidelines section. A number of tools exist for automatically converting XML DTD's to XML Schema's have been to found to be useful in performing this migration, one of which is 'dtd2schema' (www.dtd2schema.com).

#### ~~5.10~~5.6.2.3 XML Design Procedures

~~1)~~Determine location(s) of interface(s) between legacy system and the outside world. Interfaces should be well defined, and contain the data moving in and out of the system.

~~2)~~Capture and document any data structures, file formats, SQL calls, objects, layouts, ERD's, etc. that may be present at that interface. If the data structures at the layout are time-varying or sequenced, then document that as well.

~~3)~~Call out any standards used by data structures or file formats above and check XML registries that may have already been defined for that data representation or file format. First, check the COE XML Registry, URL: http://diides.ncr.disa.mil/shade/index.cfm, then check other sources like www.xml.org for XML DTD's or Schema's. <Prefer XML Schema's to XML DTD's.> For example, several standard file formats already have XML Schema's specified for them. Do several web searches using Google or other Internet search engine to look for keywords and concepts embodied in the data representations.

~~4)~~If a standard XML Schema has been identified, map your data structure, file format, etc. to that schema. If no standard XML Schema was identified, then arrange the data structure in as hierarchical an arrangement as possible. Note that many relational database products now come bundled with tools that will automatically convert a database schema into an XML Schema. However, this does not always apply when one needs to map a database to an externally defined or standard XML Schema. Use a tool like XMLSpy to create a custom XML Schema when an existing one cannot be located.

~~5)~~Once XML Schema's have been defined for each interface, gather them together and review them for commonalties that may provide the opportunity to share a schema type that differs only slightly between one interface and another.

~~6)~~Create several, if not many, different sample data files and validate them against the newly minted XML Schema using a tool like XMLSpy to verify that the schema actually reflects what is wanted.

~~7)~~Let your developers work with the schema, reading and writing the data at the interface as XML files to make sure that it is straight forward to access from a programmer's point of view. Sometimes, a small change in the schema can make writing the code much simpler. Also, take the opportunity to review the schema for redundancy in optional features, and specifying cardinality. Make sure required things are really required. Are these attributes really necessary, or would they be better as elements.

~~8)~~Upon completion of the above step, you have an XML Schema definition and some sample data files to hand over to the development team <which will appreciate you spending the extra time to develop a good specification.>

~~9)~~Once the XML Schema has matured, submit it to the DoD XML Registry, cited above, to promote re-use. Continue to look at new XML Registry schemas, as they become available, for opportunities to merge emerging existing Schemas using the process improvement methodology of "best practices".

#### 5.6.2.4 Web Services and Legacy Applications

The following bullets describe a top-level approach to create Web Services from legacy applications, summarizing aspects of sections 5.1 through 5.4. RETAINING THIS WRITEUP IN SECTION 2.0 MAY HAVE VALUE

~~2~~Identify key functionality that is required to support existing and future users of the application.

~~2~~Decompose the application into discrete services that provide the functionality identified above.

~~2~~For each service, search the enterprise registry to determine if similar service already exists and is suitable.

~~2~~For each service, define the SOAP/XML request/response messages.

~~?~~ Implement a module that accepts the SOAP request message, performs the indicated service and generates the SOAP response message.

~~?~~ Test the service

~~?~~ Submit the module to the appropriate registration authority for publishing in the enterprise service registry. (Registration authority will conduct testing and verification prior to publication).

~~**TFWeb provides steps for the process of phasing out a legacy application. TFWeb specifies different level of integration for applications and services. If an application is to be phased out, but it is desired to TFWeb enable the application in the interim, then the application should be integrated in a minimum manner. This would mean Level 1 or Level 2 integration. Level 1 and Level 2 integration require the least amount of development and, therefore, the minimum amount of investment in an application that is do for replacement.**~~

~~**Goal:** Navy Systems that are Enterprise Domain Information Technology Solutions, employing XML/SOAP interfaces within and between Enterprise Domains~~

~~**Level 3 Integration Solution:** XML/SOAP-Enabled Navy Legacy Systems~~

~~**Most Navy legacy systems require modification of interfaces to make them meet TFW and NMCI requirements. By XML/SOAP-enabling these systems to Level 3 TFWeb integration, these systems should be able to interoperate on the TFW portal in an NMCI environment. The steps required to achieve Level 3 integration are described in Section 2.4.**~~

## ~~5.7~~5.7 ~~Method for~~ Migrating and Retiring Navy ~~Replacing~~ Legacy System(s), Replacing them with Navy Enterprise Domain Solutions

This section focuses on implementing the Long Term Goal: Navy Systems that are Enterprise Domain Information Technology Solutions, providing web services and employing XML/SOAP interfaces within and between Enterprise Domains.
~~**Replacement Solution:** Migrate and retire Navy Legacy Systems into Enterprise Domain Information Technology Solutions, employing XML/SOAP interfaces within and between Enterprise Domains~~

Some Navy legacy systems may require replacement to comply with NMCI and TFW requirements. ~~Navy legacy systems,~~ XML/SOAP-enabled or not, Navy legacy systems in most cases represent stovepipe~~ systems~~s that duplicate both functionality and data with other systems, both within an Enterprise Domain and between several of them …for the same or different purposes. Business and system architectural changes may be required to evolve one or several of these Navy legacy systems into a next generation system that is ~~a~~ an effective ~~good~~ Enterprise Domain Information Technology Solution providing web services. This section describes the steps required to migrate and retire one or several Navy legacy systems into a Navy Enterprise Domain System employing an object oriented design to provide web services. Figure 5-1 provides an example of an Enterprise Domain Application Object Model

The following graceful approach supports users migrating from legacy systems, facilitating legacy retirement. It also provides the framework and foundation to facilitate the phased evolution of Navy Enterprise Domain Systems to replace one or several legacy systems. The approach entails building interfaces to the legacy system to be replaced and operating in parallel until interfaces are built to bypass it. At that point, the user can retire the legacy system at any time. Connectors and APIs will be used as standard mechanisms for interfacing with Navy Enterprise Domain Systems during legacy system migration and for maintaining the long-term interfaces with corporate systems. To reduce complexities and the associated development costs of implementing multiple interfaces simultaneously with legacy systems, a system-based replacement strategy is preferred rather than a function-based strategy. In situations where multiple legacy systems are to be migrated, priorities are set by business decision.

The example in Figure 5-1— Legacy System Migration Approach, N1 Example, uses four phases for legacy system retirement. The number of phases used in a particular legacy migration depends on the complexity and scope of the target system. Phases on the graphic are numerically labeled.

The first phase entails providing interfaces to port data (receive) from the legacy system. During the second phase, interfaces are provided back to the legacy system to exchange (send) data. In the third phase, two-way interfaces are established with systems that are dependent upon the legacy system to be replaced. Lastly, in phase four, the Navy Enterprise Domain System is validated and the legacy system is retired.

Referring to Figure 5-1, key aspects of phase one are listed below.

~~?~~ Navy Enterprise Domain System data transformation services dynamically convert data (format and values) from the legacy system.

~~?~~ Navy Enterprise Domain System data transformation services make use of a thesaurus (meta-data repository) to maintain all legacy system mappings to the internal system format.

~~?~~ Brokerage service provides a layer of transparency and manages data from multiple sources.

Key aspects of phase two are listed below.

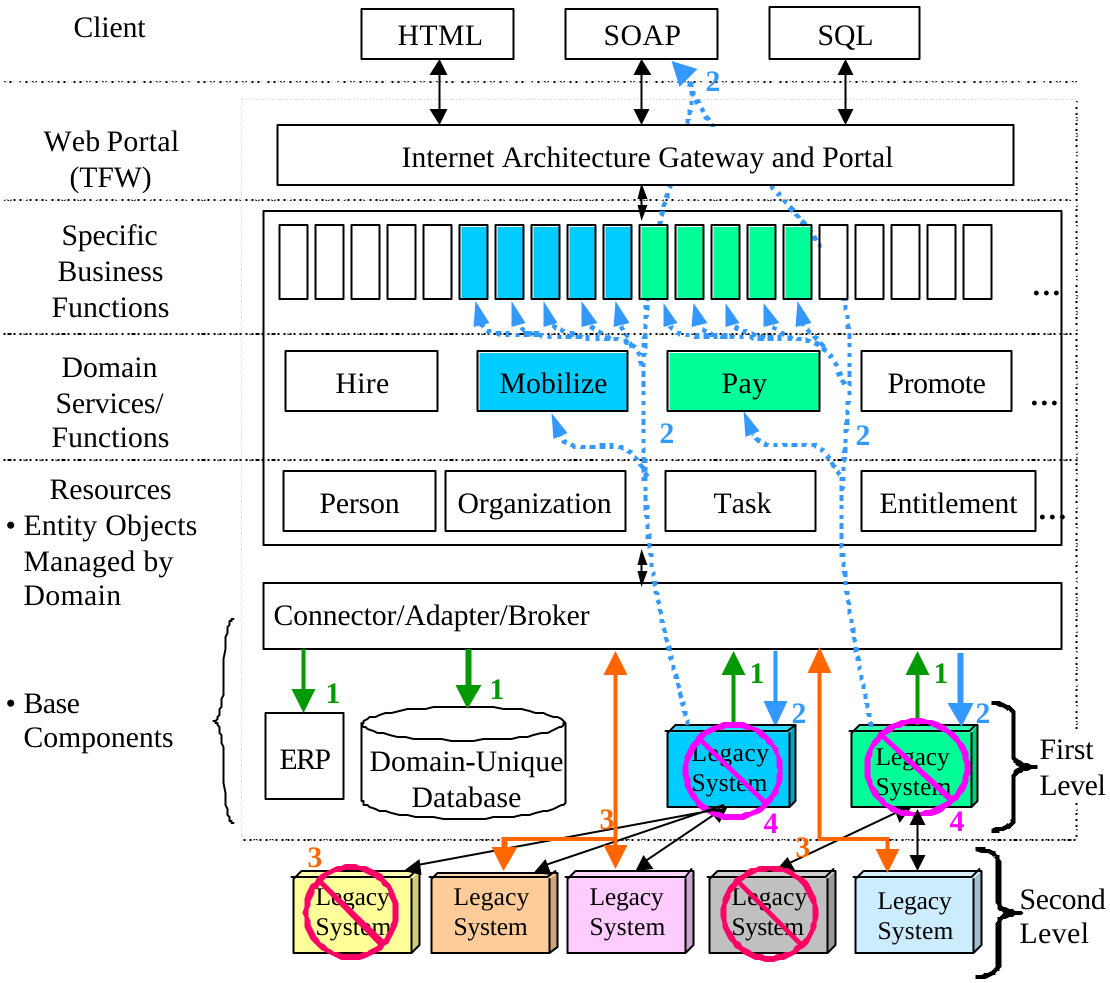
~~?~~ Business process-reengineered legacy system functionality is developed and deployed in the Navy Enterprise Domain System.
~~?~~ Connectors are developed to interface back to the first level legacy systems.
~~?~~ First-level legacy-system users are trained and migrated to the Navy Enterprise Domain System user interface.
~~?~~ The Navy Enterprise Domain System uses its broker and the legacy system connectors to pass information to the legacy systems in a manner they expect.
~~?~~ Legacy systems will be able to keep their second level external interfaces vibrant, while the Navy Enterprise Domain System develops new interfaces to these dependent systems.

Key aspects of phase three are listed below.

~~?~~ Two-way connectors are developed to interface to second-level dependent legacy systems which perform OLTP.
~~?~~ Users are trained and migrated off second level OLAP systems.
~~?~~ The Navy Enterprise Domain System uses its broker and legacy system connectors to pass information to the first and second level legacy systems in a manner they expect.

- Key aspects of phase four are listed below.

~~?~~ The Navy Enterprise Domain System is fully operational with migrated users and functionality.
~~?~~ The first level legacy systems are fully retired.



1. **Obtain data from Legacy System.**

2. **Move Legacy System Business Rules into new Object-Oriented System. Move User Interface to Web Portal. Enable two way data exchange with Legacy System, running in parallel.**

3. **Build Brokered Interfaces for Legacy External Interfacing Systems. Kill off unnecessary data warehouses**

4. **Services turn off Legacy Systems**

Figure 5-1 —Legacy System Migration Approach, N1 Example

## 5.8 Service Registration

The Service Registration package is submitted to the AMCS contact after development has been completed. The package template is available at URL: NEED URL. AMCS performs the following items as part of the package review and, as such, the Service Registration package must pass the following described scrutiny. This section also applies to changes required as part of the beta testing procedures prior to restarting testing.

### 5.8.1 Verify completeness and accuracy of portal metadata

Portal metadata should include the directory entry text, category, description, application owner, and application “customer service” contact. This is information available to any user of the portal. Is it sufficient to determine whether access to an application is required and how to obtain access? Does it address intended user base and purpose of the program? Similar programs directed at other user bases should be mentioned in the description.

### 5.8.2 Verify migration plan for level of integration is submitted

Migration plan is required for applications migrating at Level I or Level II. For the pilot program, the application may enter beta testing prior to submission of a migration plan. However, the TF Web Governance Board will not approve integration into the production portal without a migration plan. A migration may be as brief or as detailed as desired; however, timeframe, critical path, and issues to be resolved must be included. Retain copy of migration plan in AMCS Echelon II notes for future reference. Brief the migration plan to AMCS OIC for approval.

### 5.8.3 Ensure IATO/ATO has been updated if security model changed for TFWeb migration

Provide copy of IATO/ATO cover letter to AMCS IA for reference documentation. Submission of full accreditation paperwork is not required unless determined necessary by AMCS IA.

### 5.8.4 Verify initial access control list submitted along with information describing method of updating ACL

Verify method is compatible with current portal capabilities and user expectations. If access cannot be given in a timely manner, ensure it is indicated in the application description visible to the user. Review and approve appropriate roles for application visibility. For most applications that control security at the application level, the ACL should be “All Portal Users”.

### 5.8.5 Portal Compliance Testing

The AMCS liaison shall be provided a temporary login with access to key features of the application. In the event access to key areas cannot be provided due to security/access issues, alternate methods will be coordinated between AMCS and the application developer. Spot check to ensure claimed capabilities of user description are provided and significant limitations are documented. Spot-check HTML used is "portal compliant" (no frames). Record all concerns, discuss with developer or program manager. Submit any unresolved discrepancies to AMTS (for pre-beta review) or as part of the AMCS beta testing notes. This is not intended to be a thorough review of the program. Rather, it serves to ensure any obvious issues are recognized and documented prior to the beta testing process to help expedite testing

### 5.8.6 Review summary of testing accomplished

Summary should include duration, type of users, type of test scripts performed, type of data used, and environment in relation to the production platform. What is the risk that the program will fail beta testing? Has there been sufficient operator testing to ensure utility in the production environment?

### 5.8.7 Review portal integration information submitted

To ensure effective use of testing time and to allow maximum preparation time for testing, all required portal integration information should be submitted as part of the request. While further changes may be necessary or desirable, this allows for a package of all required information to be submitted from AMCS to the beta test site. Ensure integration module code has been provided.

Identify if substantive revisions have been made to sample code. Module code should be fully documented and readable. Evaluate code for posting to open source site (based on differences from baseline code). If review of code is required, submit request to AMTS. Are there any reusable components that should be separately maintained?

### 5.8.8 DoN XML guideline compliance

If the application does not meet DoN XML guidelines, ensure migration plan is submitted and approved by AMCS OIC.

### 5.8.9 Set next review date

General guideline is 1 year if all requirements met or halfway to next integration level (3 month minimum) if a migration plan has been submitted for non-Level 3 integration or noncompliant XML are used. If IATO has been submitted, review date should be prior to its expiration. By this review date, a member of AMCS will review documentation and implementation history, and determine if any additional information is required. Milestones in migration plans or further functionality development will be reviewed. Also, database information will be verified.

### 5.8.10 Verify database entry is complete and accurate in AMCS application database

AMCS database information is in section, Submission to the application information database.

### 5.8.11 Technical Review

AMCS may request a technical review at any time from the AMTS or alternate source. This technical review may evaluate code base, technology, mobile code, or security among possible areas. In some cases this is used to evaluate leading-edge technology and possible unforeseen impacts on TFWeb environment. In other cases, it is used to check for compliance with TFWeb architecture in a more thorough fashion than is possible in the beta testing environment. If necessary to evaluate an application's readiness for migration, this review is completed and any discrepancies resolved prior to permission for beta testing. The AMCS is the final arbiter of whom discrepancies are required to be resolved prior to beta testing, though review of an AMCS decision may be requested from the TFWeb Executive Steering Group.

### 5.8.12 Configuration Verification

Verify configuration of any local application servers or local remote module servers are documented. The ability of local infrastructure to support numbers of users intended should be documented as well as ability to scale to additional users. Any known scalability issues should be documented.

### 5.8.13 Ensure application is logged in the DON CIO Data Management and Interoperability Repository.

DMIR is currently in the beta testing stage. This is an optional requirement until full functionality in the second quarter of 2002.

### 5.8.14 Verify all application data structures and data interfaces are documented.

Databases should be accessible independently of application if underlying database engine and security supports. Data interfaces should also be accessible independently.

### 5.8.15 Verify AMCS OIC has approved migration p lan for application/data overlap.

Migration plan should address duplicative applications and data sources and their planned resolution. Migration plan is not due until final review of application after beta testing.

### 5.8.16 Documentation of Developer Requirements.

Ensure developer requirements for future capability upgrades of the WEN architecture and implementation of the architecture are documented. This should consist solely of architecture or cross-application services, not those useful only to a single application. This helps prioritize additional requirements based on the ability of the developer community to capitalize upon the new features. This should only include functionality developers are currently able to utilize.

## 5.9 Application/Service Delivery Phase

### 5.9.1 Application Acceptance

Only applications that have completed the migration request process with TFWeb are submitted (e.g., application components, links/icons, datafill, DTDs/Schemas) for integration in Enterprise Portal.  All applications need to ensure compliance with required DoN/DoD policies in addition to those required by TFWeb. The TFWeb process does *not* supercede individual program, application, or content manager processes.  It is expected that the Enterprise Portal will receive applications that have gone through internal system engineering and logistics processes (e.g., CCB, internal testing, CM).

### 5.9.2 Application Delivery

Each application is expected to deliver system and administration documentation that conforms to Enterprise Portal documentation guidelines (e.g. XML or HTML).  This includes software operation and concise loading instructions to enable users/administrators to load and administer the applications with minimum intervention. The instructions should also include load verification and load back-out procedures.

Once the application is successfully loaded into the TFWeb developmental portal environment, the developer will then continue with the remainder of the self-certification procedure, moving into the performance criteria.

### 5.9.3 Application Integration

The developer is encouraged to notify TFWeb of an impending application release no later than 30 days prior to portal integration.  This gives the TFWeb team time to arbitrate schedule conflicts with other application developers.

The application integration process differs depending upon the type of application to be integrated, and the level of integration the application is achieving.  In all cases the goal is to provide the developer a process and supporting infrastructure by which they can develop, test, and certify their application(s) for use in the Enterprise Portal environment with a minimum involvement by a core TFWeb team or other external agencies.

# 6.0 Information Assurance

## 6.1 Strategic Overview

Within the military community, Information Assurance (IA) and Computer Network Defense (CND) are the two defensive areas that have been incorporated under the greater Information Operations (IO) umbrella. In this broad threat environment, where every connection to a network must be regarded as a potential avenue of attack, IO must defend not only our own information and information systems, but also affect adversary information and information systems to deny their capability to be utilized against us. To do this, IA supports the full-dimensional protection aspect of Joint Vision 2020 and comprises actions at the tactical, operational, and strategic levels that protect and defend information and information systems by ensuring ***availability, integrity, authentication, confidentiality,* and *non-repudiation.*** In addition, Joint Pub 3-13 *Information Operations*, stresses the importance of IA in the overall success of Defensive Information Operations (D-IO) based on four interrelated processes: attack protection, attack detection, restoration and response, as well as by coordinating interoperability among the services and coalition partners, as well as government and non-government organizations.

IA seeks to insure the security of information in its myriad of forms, not just information transferred using telecommunications or stored computers. Thus there is a close, synergistic relationship between counterintelligence, operations security, communications security, information security and information systems security all of which seek to protect information from hostile access and exploitation. This concept is much more expansive in scope though than classic information systems security which many people normally tend to relate to. It encompasses those communications and computer network management functions that seek to provide for continued operations in the event of accident, natural disaster, deliberate act, and adverse operational environment.

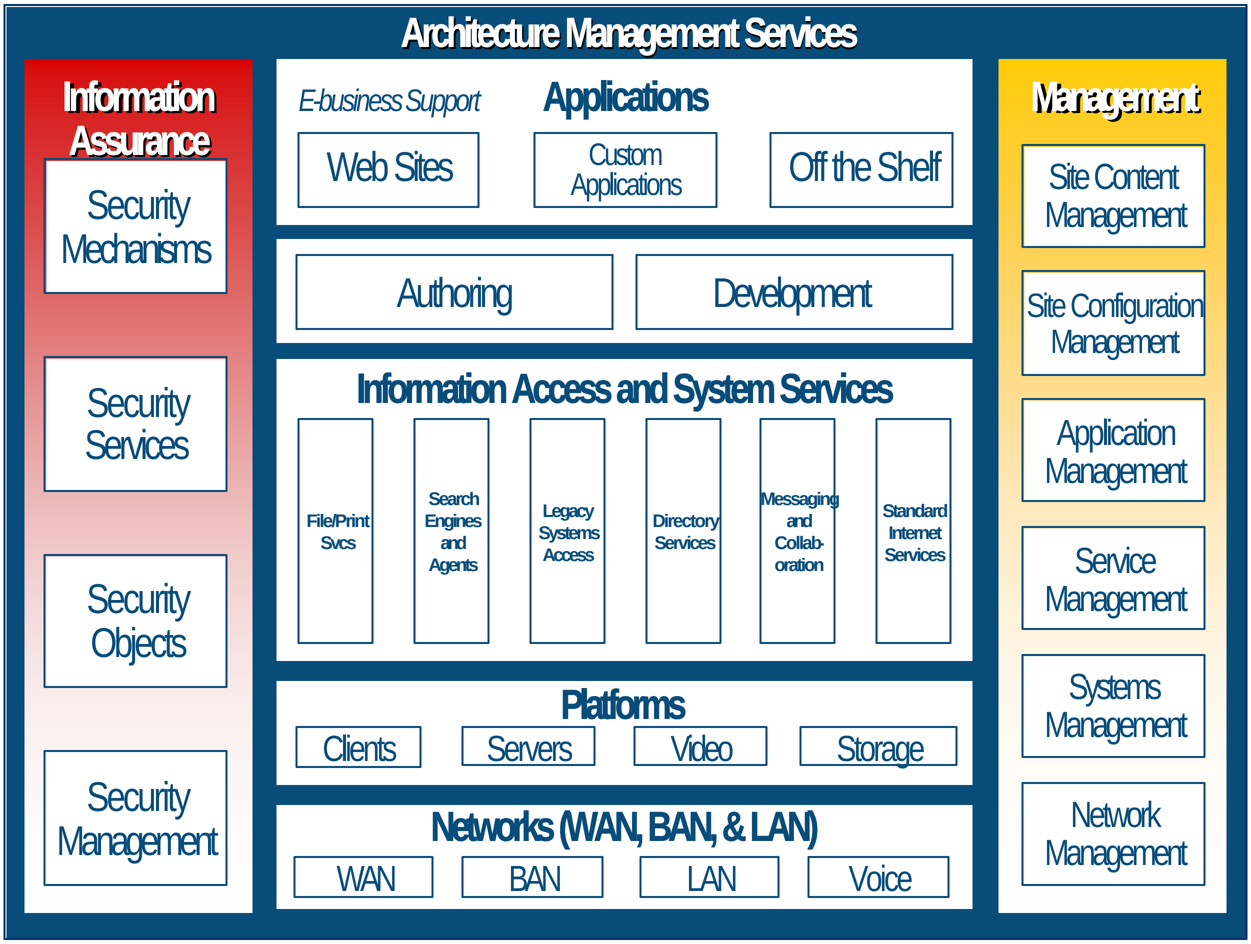
One can readily understand how information security considerations are critical to all DoN information systems, including the new Navy Enterprise Portal. The open nature of modern information systems provides hackers with many avenues of attack; therefore all C4I systems must employ a "defense in depth" strategy to protect mission critical data and services. The application portal being designed by Task Force Web is thus expected to play an important role in the enforcement of the WEN (Web Enabled Navy) security policy and therefore must be capable of reliably delivering fundamental information security services.

## 6.2 TFW IA Overview

The Navy Enterprise Portal will service the needs of the Navy's business and operational user communities. In order to accomplish this goal, IA as described above must therefore be incorporated throughout the entire architecture of the system. Separate instances of the portal will be implemented on the Non-Classified Internet Protocol Router Network (NIPRNET) and on the Secret Internet Protocol Router Network (SIPRNET) in order to support both unclassified and classified processing requirements. The portal must meet the minimum criteria specified for connection to these networks. A single solution is required for both unclassified and classified environments. The components of the portal will be located in facilities with appropriate physical protection controls based on the classification level of the data being processed. Portal implementation requirements at the TOP SECRET level are currently To Be Determined (TBD).

The portal is required to operate in an environment where all portal users are approved (cleared) for access to all data served by the portal; however, all users will not have a legitimate reason (need to know) to access all information served by the portal. The portal will operate in the “systems high” security mode of operation, enforcing a discretionary access control security policy using mechanisms equivalent to Class C2: Controlled Access Protection (as currently defined by DoD 5200.28-STD.) As the portal product market matures, products that have been evaluated based on the Common Criteria will become available with the adoption of DoDI 8500. When this occurs, preference will be given to products that have achieved a minimum of EAL 3.

All DoD information systems are required to be formally Certified and Accredited (C&A) prior to being placed in operation. The Navy Enterprise Portal is currently being certified and accredited in accordance with the DoD Information Technology Security Certification and Accreditation Process (DITSCAP) as defined by DoDI 5200.40. A formal certification and accreditation plan has been developed that addresses both the accreditation of the portal infrastructure and the applications served by the portal. Application providers/developers will be required to provide proof of independent accreditation before submitting an application for incorporation in the portal.



This diagram shows how crucial IA is to the success of NMCI. Through layers of technical protections and procedures, NMCI enables its users to access information and services with the trust necessary to do their jobs. Defense-in-depth protection mechanisms are deployed in a layered fashion forming boundaries at multiple levels within the security architecture. This process ensures resistance to attacks and minimizes the possibility of a security breach due to a weakness (known or unknown) at any single security component. The defense-in-depth protection strategy provides security features to NMCI systems and data. These features are confidentiality, integrity, availability, accountability, and non-repudiation as mentioned earlier, and the TFW portal will allow NMCI to truly comprehend it's potential. However the actual realization of the Navy Enterprise Portal's full potential is dependent on the availability of infrastructure services that are outside the scope of this document. Among these is the availability of the DoD Public Key Infrastructure (PKI). The objective portal system must fully support the DoD PKI when it becomes available.

The portal shall, as a minimum, support the following fundamental information security services:

Authentication - a means to establish the validity of a claimed identity. The user's identity can be verified as part of the certificate-issuing process (literally, the user is authenticated).
Confidentiality - is assurance that information is not disclosed to unauthorized persons, processes, or devices. The process of assigning rights, which includes the granting of access, is based on specific access rights. The authorization policy is a control policy by which access by subjects to objects is granted or denied. An authorization policy will be defined in terms of access controls lists, capabilities, or attributes assigned to subjects, objects, or both.
Integrity - is a security service that protects information from undetected modification.
Availability - the state when information services and/or data are in the place needed by the user, at the time the user needs them, and in the form needed by the user
Non-Repudiation - provides undeniable proof of a party's participation in a communication.

The functional requirements in the following subparagraphs have been organized based on these five fundamental information security services. The functional requirements described in this document must be interpreted within the framework of existing DoD and Navy security policies. Current DoD/DoN policies such as the Mobile Code policy and Fleet Firewall policy will evolve as web technology continues to change.

### 6.2.1 Authentication

The portal shall identify and verify the identity of an eligible user (not just a person, this could be any device capable of using a PKI certificate). Authentication to the portal shall require presentation of a valid Class 3 or Class 4 PKI Digital certificate, along with a pass phrase or equivalent "something I know" validation; or of a password associated with a unique user ID as outlined in the *Web Server Protection Profile*, National Security Agency, draft January 2000. The requirements are as follows:

Strong authentication shall be required for any user access to the portal, and all such accesses shall be audited.

Error feedback for user authentication shall contain no information regarding which part of the authentication information is incorrect.

The portal shall limit the number of unsuccessful login attempts. The number of unsuccessful logons shall be configurable by an administrator.

The portal shall have the capability to limit the number of concurrent logons. The number of simultaneous logons shall be configurable by an administrator.

The portal security architecture shall eventually support the use of "single sign-on" (SSO).

### 6.2.2 Confidentiality

The portal shall support confidentiality. The portal shall provide a centralized mechanism to enforce access control at or above the object level (i.e., functions, data) based on a subject's (user, applications) valid identification, authentication, roles, and permissions. The portal shall provide one or more mechanisms to permit applications to ensure confidentiality of sensitive transmitted data via data encryption using government-approved means in accordance with appropriate PKI policy. Approved Secure Sockets Layer (SSL) methods shall be used to provide confidentiality of the data in transit. The portal shall provide a capability for centralized user account creation in a heterogeneous environment with the capability to define a unique user identifier and login name (within administrative domain). The capability to define user profiles shall be capable of supporting central storage and central management of user profiles. The capability to define user profiles shall support the definition of which system functions, systems, applications, and files a user with a given profile shall be authorized to access.

### 6.2.3 Integrity

The integrity of the portal shall be maintained in accordance with standard industry IA practices. The data shall be protected to ensure that the information while in transmission and in storage has not been altered. The use of SSL shall be used to preserve the integrity of the information while in transition between the client and the server. Web servers shall be securely configured as outlined in the *Web Server Protection* Profile, National Security Agency, draft January 2000. The secure server configuration shall include the ability for all web servers within the portal to use PKI certificates and certificate services that are part of the DoD PKI. Web servers shall be securely administered as specified in the *Department of Defense Web Administration Policies and Procedures*, 25 Nov 1998, published under ODSD memo 7 Dec 1998. Restricted access web servers shall implement secure Web technology (e.g., SSL/PKI) as mandated in the *DoD PKI Memorandum*, 12 Aug 2000. Restricted access servers shall employ Class 3 or 4 digital certificates issued by the DoD-PKI to perform authentication. The use of Mobile Code will comply with relevant Mobile Code policy, of which the most current is the Fleet Firewall Policy, DTG 30 November 2001. See the reference section for more details.

### 6.2.4 Availability

The portal shall be available in a timely and reliable manner in accordance with proper confidentiality and integrity requirements outlined in the security policy section of this document. Availability may be demonstrated in several ways including but not limited to; replication of critical portal system components, redundancy, load balancing, performing system backups, and having a demonstrable disaster recovery plan. Denial of Service (DoS) or Distributed Denial of Service (DDoS) to the portal is a serious and likely threat. The portal must demonstrate mitigation techniques to minimize the unauthorized destruction, modification, or delay of service resulting from these or any other type of attacks.

### 6.2.5 Non-Repudiation

The portal shall ensure non-repudiation using digital signatures based on DoD PKI Class 3 or Class 4 certificates and keys. SSL/PKI shall be used for non-repudiation as assurance that the sender of data is provided with proof of delivery and the recipient is provided with proof of the sender's identity, so neither can later deny having processed the data.

### 6.2.6 Accountability

Accountability is the result of providing documentation on system and/or user activities via secure audit logs within a reasonable amount of time. The ability to log and report security events, such as system access or the execution if a portal resource, shall be made available to the system administrator. Auditing shall be made available for each of the components within the portal. Audit reduction tools and audit-reporting capabilities should be available for review. The requirements are as follows:

The portal shall provide a mechanism to capture audit logs for selected actions deemed necessary by the system administrator, in order to provide him/her the ability to reconstruct events and determine individual responsibility for security related issues.

The audit mechanism shall be capable of automatically collecting, processing, and identifying security-relevant events that meet security audit requirements. Minimum auditing requirements will include: user logon, user logoff, user actions to open, close, create, delete, modify, execute programs or files, and the attempts to access protected objects (e.g. configuration files, audit files, password files, etc.).

For each audit event the audit mechanism shall record the following information: date and time of the event, the unique subject identifier (user-id) on whose behalf the subject program generating the event was operating, type of event, success or failure of the event, origin of the request (e.g., terminal ID) for identification and authentication events, name of program or file introduced, accessed, or deleted from a user's address space.

The system shall provide end-to-end system and user accountability for all relevant events so that the system administrator will be able to reconstruct the cause of an event and identify the user or system component responsible for the event.
The access and transmission audit logs shall be strictly controlled to maintain integrity.

## 6.3 Certification and Accreditation

The portal security solution shall ensure that the portal can be operated at “an acceptable level of risk” based on the minimum clearance of the user and the maximum classification of the data processed when subjected to the DITSCAP process. The Designated Approving Authority (DAA) appointed for the portal shall determine the “acceptable level of risk”. In accordance with OPNAVINST 5239-13, all US Navy information systems will be certified and accredited using the DOD DITSCAP policy. For the TFW portal, all applications will route their IATOs to the TFW ISSM..

## 6.4 Security Policies

The portal will be used to access sensitive information therefore; it must be designed to guarantee the correct and accurate interpretation of all relevant security policies. The portal will provide assurance that all relevant security policies have been followed by demonstrating derived security compliance in the system architecture via security testing, configuration management, design documentation and user guides. All portal implementations for these critical elements must comply with these fundamental Information Assurance policies and practices documents, listed in the Reference Section.

The TFWeb architecture will follow an 'Allow then Deny' approach to security for the Portal and backend Applications, i.e., all of the services in the Portal are visible to (nearly) all of the users. A minimum number of Portal workgroups will be created to define which Portal connectors certain user groups will be able to see. Each group will define a large number of connectors that the users will be able to see. Based on DoN policy, there may be a very few exceptions (in case of very high security applications), where the Portal connectors will only be visible to a few authorized users. Thus, the majority of the portal connectors will be visible to all the workgroups in the Portal.

Portal users can arrange the Portal connectors on their workspace of choice by dragging and dropping them into their display. At this point in time, the Portal connector will try to connect to the underlying Application by sending an HTTPS request to the Application’s URI on the EMS server. The SSO web server agent on the EMS server will then either allow or deny access based on the user’s rights to the Application as defined in the Entitlements Database. Thus, in some cases, even though the Portal connector is visible to the end user, the actual contents of the Application may not be available. The user will need to request access to the Application from its owner.

In case of an Application’s Portal connector, access control is achieved through the security policies set in the SSO server. The SSO web server agent installed on the EMS server will challenge any requests for Applications and allow or deny access based on the security policies set for the Application. It is the Application owner's responsibility to provide the necessary information to the directory and SSO administrators to set the Access Control List (ACL) policy for the Application. This information will consist of the Application's name, and the definition of all users that require access to the Application. This definition of users will be based on user's names. Once the authorized users are defined, the directory administrator will create an Active Directory group that, in conjunction with the SSO product, will allow access to the Application.

The entire TFWeb architecture, from the Portal user's browser, to the back-end Application web server, shall utilize encrypted communications over 128-bit SSL. For HTTP communications, this will be by means of HTTP over SSL, port 443. SSL shall be implemented by means of DoD PKI Class 3 certificates on both the NIPRNET and SIPRNET architectures. DoD PKI Local Registration Agents (LRA) are currently readily available to issue PKI server certificates on the NIPRNET. Application developers/owners who need assistance in acquiring a DoD PKI certificate on the SIPRNET should contact their TFWeb AMCS representative or the TFW ISSM. Of particular importance to Application owners in the interface between the EMS and their Application web server. For all levels of integration, this interface must support at least one-way (server-side authenticated) SSL communications. Based on the needs of the Application, the developer may choose to implement a two-way SSL (client and server authenticated) communications path between the EMS and the Application web server. Client-authenticated HTTPS communications will be required by all shore users to enter the NMCI TFWeb enclave and access the TFWeb Portal. This will require that all shore Portal users have a DoD PKI issued identity certificate from the Class 3 PKI. Therefore all Users should contact their command's LRA or their TFWeb AMCS representative for assistance in getting the required PKI certificates.

### 6.4.1 Single Sign-On Security Architecture

The SSO solution for the TFWeb architecture:

Acts as a central point of authentication for Enterprise Portal users,

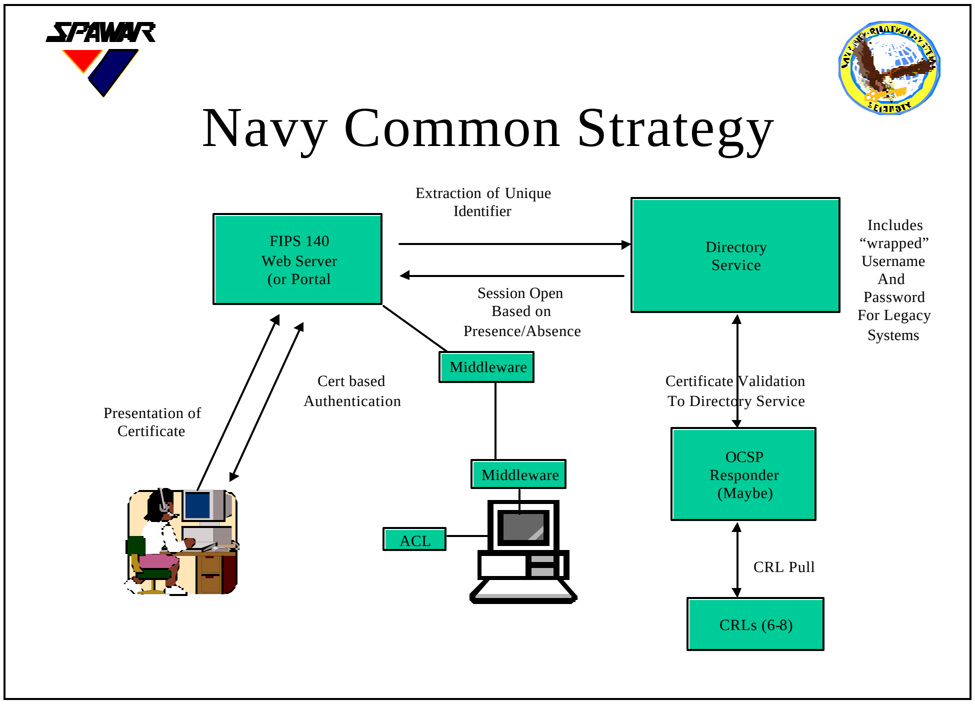
Provides a framework for authenticating users which will enable a Navy-wide web SSO implementation by Application owners,

Authorizes the use of Portal services and Applications based on dynamic security policies configurable by Application owners,

Utilizes the enterprise Active Directory architecture by replicating users, user attributes, and security groups to establish Portal security policies.

The SSO solution provides a single, unified mechanism and interface for controlling access and security across platforms, applications, and Web servers. More specifically, SSO authentication and authorization solution provides centralized management across platforms and vendors, delegated user management, rules-based access control, and support for multiple forms of authentication. Single Sign On is defined as the process of a user logging in to a system only one time, and thereby having access to all resources for which they have rights, without having to log on to those resources individually.

The ultimate goal therefore of the SSO solution is to provide a framework to allow Navy Portal users to move seamlessly across the Portal web servers and Applications without having to re-authenticate each time they click a new link. Their authentication information is passed on to other SSO components via an encrypted temporary session cookie. SSO depends on the storage of authentication information in the encrypted temporary session cookies. At runtime the SSO web server agent communicates with the SSO Server, which can encrypt and decrypt the temporary session cookies. Therefore, a SSO web server agent can always decrypt a cookie generated by itself or any other Web Server Plug-in in the system, and can authenticate using stored information rather than re-prompting the user for credentials.


SPAWAR
Navy Common Strategy
Extraction of Unique Identifier
FIPS 140 Web Server (or Portal
Directory Service
Includes "wrapped" Username And Password For Legacy Systems
Session Open Based on Presence/Absence
Middleware
Cert based Authentication
Presentation of Certificate
Certificate Validation To Directory Service
Middleware
OCSP Responder (Maybe)
ACL
CRL Pull
CRLs (6-8)

.

#### 6.4.1.1 Role of the Directory in SSO

The enterprise-wide TFWeb Active Directory architecture is the basis for all Navy Portal SSO operations. The TFWeb directory architecture will be replicated/synchronized across the Navy enterprise, and will be collocated with the Portal architecture components. This directory architecture will also extend to include Navy afloat ship platforms. The primary function of the TFWeb Active Directory architecture is to provide a centralized location for the authentication and authorization of all Navy and United States Marine Corps (USMC) users to the TFWeb Portal architecture. In order to be the centralized source for all user authentication, the TFWeb directory will establish global unique user identifiers for all registered Navy and USMC TFWeb users, both afloat and ashore. These user identities will be based on the flat SMTP name space being fielded by the NMCI (e.g., joesph.user@navy.mil, joesph.q.user@navy.mil, jane.user@usmc.mil, jane.j.user12@usmc.mil). Every user will be assigned a new unique ID in the flat name space that will remain theirs, regardless of their location within the Navy or USMC organizations. TFWeb will use a temporary flat name space for non-NMCI users (@tfw.navy.mil and @tfw.usmc.mil) in combination with the NMCI flat name space. Authentication of users to the TFWeb Portal and its Applications will be based on the flat name space userID. The SSO server will perform authentication of a user's identity directly against the TFWeb Active Directory. The TFWeb Active Directory will also be the centralized source for authorizing user access to TFWeb Applications. Active Directory groups will be defined determining a user's ability (or inability) to use a particular Portal Application. The SSO server will utilize these groups in allowing or denying users access to the Portal and its Applications. The TFWeb directory service is the initial implementation of an enterprise-wide directory to support Navy and USMC applications, such as the WEN. The TFWeb directory is the basis for, and will eventually evolve to become, the larger Naval Global Directory Service (NGDS). The NGDS will provide enterprise-wide services such as TFWeb authentication, location of Navy and USMC personnel (Navy/Marine Corps White Pages), and convergence of NMCI and IT-21 directory services into a logical global directory

#### 6.4.1.2 Current SSO Issues

DoD is in the process of issuing Class 3 (medium assurance) PKI certificates to servers and individuals. TFW intends to utilize these certificates to provide the full range of PKI Security Services, one of which is stronger authentication. In addition, the following questions are also being addressed in order to develop a long term SSO architecture:

Also need assurances and specifics on certificate validation (e.g. OCSP/CRLDP validation of DoD PKI Certificates).

Relevant to the function of the key server, what specific cryptographic functions are being performed by the key server, and are they currently using FIPS certified products to perform these functions (e.g. BSAFE or other FIPS certified). If DoD PKI certificates are used and all backend and front-end servers have DoD PKI certificates, what is the role of the key server?

Is server-to-server two-way authentication used between front end and back end? Relevant to the function of the PKE Agent, are proprietary protocols utilized for agent-to-agent (front end to backend) communications? If so, why does the vendor consider that they are necessary?

What standards are being applied in front-end to back-end connectivity and information transfer (e.g. CORBA, RMI, other)?

What specific information is contained in the session cookie/http header?

What is their strategy for capturing legacy non-web applications, and do they provide any toolkits for doing this?

Relevant to firewall friendliness: What specific ports are required to be open? Can the implementer modify this? Has the vendor requirement for opening ports been minimized?

In the event that certificate based authentication is implemented, is there any means to manage access (authorization) based upon the strength of credentials presented at the authentication server (e.g. a hard token is a stronger means of authentication than username/password - can authorization for a single user be applied to reflect the strength of the authentication token, or is this a one-size-fits-all?).

What parts of the architecture have gone through FIPS certification? What is the current strategy for FIPS (NIST) certification? Same question for Common Criteria.

While the theory of how SSO can benefit TFW is good, the reality is that technology to support this architecture is still being developed. There are essentially two means to overcome this shortfall.

Accept the current product limitations and continue to execute the necessary “work-arounds” in order to make the product work, based upon the assumption that promised vendor product improvements will eventually achieve compliance and interoperability.

Pursue a different solution that meets the immediate requirements for compliance and interoperability.

#### 6.4.1.3 LSSO Introduction

As an interim solution to the single sign on architecture, Lightweight Single Sign On (LSSO) describes a simple method to transfer the flat name UserID to the application or service by using the existing XML PRI interface to provide UserID and SessionID directly to the service or application. This solution may provide enough user identification for most application/service developers, while pushing the developers to use the Navy Global Directory Service (NGDS) flat name space. The Lightweight Single Sign On is based on the following assumptions:

1-way SSL between EMS and Service host. (allowed by Section 6.4.1 of TFW Developer's Guide)
Consistent passing of PRI from the portal to the service.
Application/Service is at a minimum of Level 2 integration.
Understanding the risk associated with trusting the PRI header via 1-way or 2-way SSL as appropriate.

The Portal product has validated the UserID through the standard login procedure, i.e. validated by the existing SSO product and Active Directory.

### 6.4.1.4 LSSO Details

The Portal Connector will pass the PRI header to the Level 2 service module when called from the portal. Upon parsing the XML PRI in the header, the application will be able to retrieve the following information as defined in this guide: Reference the table vice putting copy of table here.

| Data Elements | Description |
|---|---|
| UserID | Portal user's ID based on NGDS flat name space schema |
| RoleAssignments | Group from directory service |
| PortalLocation | Determines if ashore or afloat |
| Client | For the pilot, set to "browser" |
| CheckBandwidth | For module to determine if there is bandwidth availability. |
| SessionID | A session identifier for the portal's browser session. |
| ClientStyle | Reference to portal style sheet, so that service can maintain the current portal look and feel. |

For LSSO, the passed UserID along with stated assumptions should be sufficient for user identification for most services/applications. It does not offer strong authentication, like commercial SSO products such as RSA ClearTrust, Baltimore SelectAccess or Oblix Netpoint; however, it will provide a simple means to a service to use the UserID. The service developer will then use the flat name as the unique identifier for the users of the service, which will make the transition to full SSO services. The service developer assumes the PRI passed from the Portal is accurate and valid and has not been spoofed. While this assumption carries significantly more risk than a full SSO implementation, many existing web applications do not require high-end authentication. Even though trusting the UserID in the PRI header is not always recommended, the use of LSSO implies the developer has weighed benefits and risks opening data access associated with this method. For added trust, the UserID passed could be verified against Active Directory to verify the user exists in the NGDS and/or the SessionID could also be validated to be a possible portal SessionID. This SessionID validation would be similar to a CRC, and will not verify back to the portal that the SessionID currently exists. Also the UserID verification and SessionID validation services currently do not exist and are not required implement LSSO. Furthermore, 2-way SSL can be used to greatly lessen the risks with this proposed LSSO solution.

TFW will provide guidance to organize and ensure the minimal amount of effort for the developer to transition to full SSO when available. The benefits and risks of LSSO are summarized in the following table. To implement LSSO, the developer must do the following.

Move the members on the application's Access Control List (ACL) to use the unique name provided by the NGDS flat name schema (preferred) or provide a mapping between an existing ACL and the flat name space (alternative).

Modify the application to be level 2 integration, so that it can parse the PRI XML header information.

Trust the information passed via the PRI to be accurate and valid and understand the risks (with 1-way SSL and 2-way SSL).

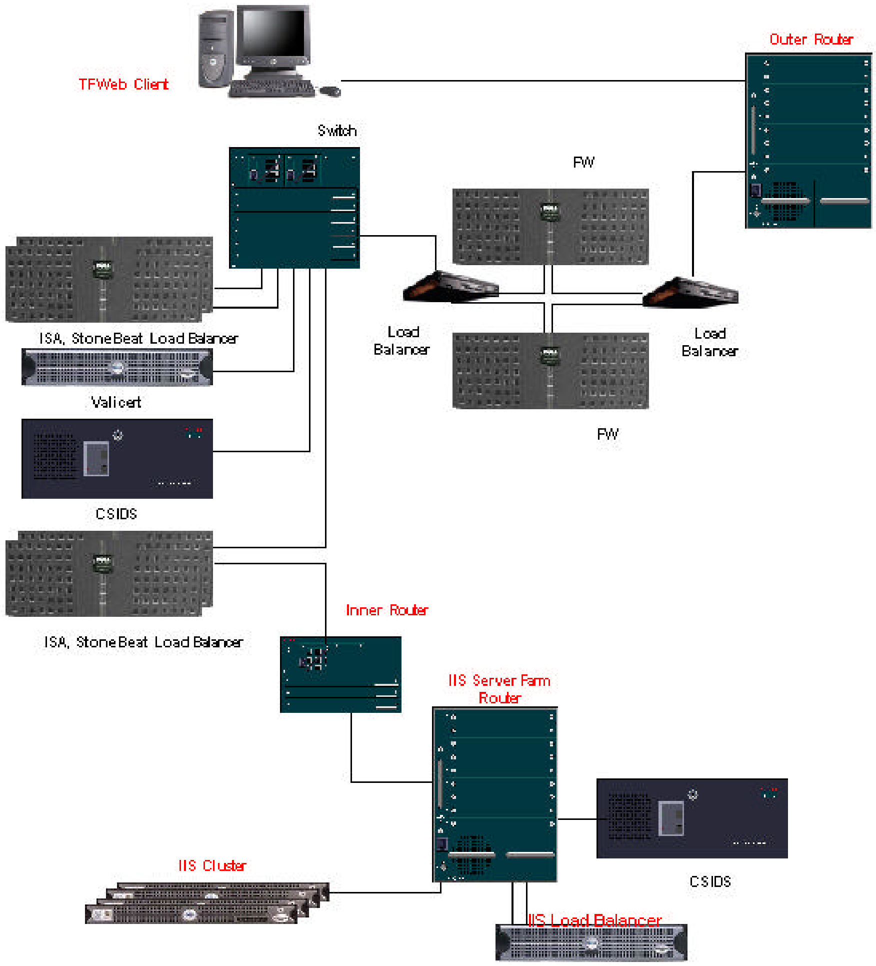
| Benefits | Risks |
|---|---|
| Migrate to NGDS flat name space. | Potential rework upon availability of full SSO implementation. |
| Access control to the service remains with the service/data owner | PRI can be spoofed in 1-way SSL solution. 2-way SSL ameliorates this risk. |
| Encourages the development of directory query services. | Modification of existing applications/services, may require re-certification of application or service. |

### 6.4.2 Public Key Infrastructure

Access is a huge issue with respect to computer security and IA. The Department of Defense (DoD) has opted to use Public Key Encryption as a preferred method of authentication. Therefore all DoD Public Key (PK) Enabled applications shall support the use of the DoD Class 3 PKI certificates and keys to provide non-repudiation, integrity, confidentiality and strong authentication as specified in Assistant Secretary of Defense (C3I) Memorandum, *Department of Defense (DoD) Public Key Infrastructure (PKI)*, 12 August 2000. The current direction from both Navy and DoD is that the Common Access Card (CAC) is the principal container for all certificates and keys associated with DoD PKI. In reality, however, due to identified difficulties with deploying the CAC to Navy afloat/tactical forces, there will continue to be a requirement to sustain the capability to issue software certificates if current DoD mandates for signed electronic mail are to be met.

With respect to TFW, a significant portion of the WENPP is to test the access of the non-NMCI User to applications via the portal. This required users to obtain and use PKI Class 3 software certificates, which requires the installation and use of the Netscape browser and Personal Security Manager (PSM). To date, TFW staff and many users have experienced significant difficulties downloading and installing these software certificates, however it is still a requirement for all TFW users to possess and utilize a PKI Certificate.

The PK enabled applications shall provide the capability to verify that a digital certificate presented by a user or system entity was issued by one of the Certificate Servers whose certificates are loaded onto the portal applications. To date, the assistance of OPNAV N614 and LRAs across the country was instrumental in arranging for rapid issuance of PKI Certs to TFW Beta users, and eventually the portal and all PK enabled applications will migrate to use Class 4 PKI certificates (i.e. smart cards or equivalent) to supplement or replace Class 3 certificates. In addition, the portal and all PK enabled applications shall have the capability of validating client and server certificates as outlined in the SPAWAR PMW 161, (May 2001), *Navy Tactical PKI End-to-End Certificate Validation Concept of Operations* DRAFT.



### 6.4.3 Non-NMCI User Access

One of the major hurdles so far for TFW from an IA viewpoint has been the incorporation of the non-NMCI user into the NMCI portal. The ISF team has come up with a solution, that uses a DMZ with two proxy servers and an IDS. The current title for Non-NMCI User Access is called the Secure Web Access (SWA). This solution will be tested in the pilot architecture to determine if there are any performance and latency issues before it is deployed across the enterprise portal.

### 6.4.4 TFW Mobile Code Use Policies

The UNCLASSIFIED information is documented in the *Navy-Marine Corps NIPRNET Enclave Protection Policy*, dated 30 Nov 2001. Please refer to the full CONFIDENTIAL policy statement posted at http://www.infosec.navy.smil.mil under Fleet Documents and Information. The UNCLASSIFIED policy can be found at https://www.infosec.navy.mil under Fleet Documents and Information.

### 6.4.5 Legacy Application Access Control Mechanisms

As stated above, SSO will provide enterprise-level security, including authentication and authorization to access the URL on the EMS server that defines the ‘home page’ for a portal Application. However, it is the responsibility of the Application owner/developer to provide local security, at the Application level. This security includes determining the process by which users are authenticated and authorized to the legacy Application. To participate in the overall SSO solution for TFWeb, Application developers must:

Determine the means by which users will be authenticated to the legacy Application through the Portal

Map TFWeb userIDs to the Application’s internal user rights database (for legacy Applications) or develop a user rights database using the TFWeb userIDs (for new Application development)

Provide TFWeb SSO administrators with a list of users authorized to use the Application (see section 6.7.1). Authorize user’s rights to internal Application processes or databases based on the authenticated userID provided by SSO.

In particular, the Application must assign all rights that are necessary internal to the Application. Examples of these user’s rights include: allowing users to modify data, providing access to portions of internal databases, and administering other user’s rights.

Application developers must also continue to provide sound local security for their web servers and Applications. This local security may include mechanisms such as applying all appropriate patches for the operating system and web server software, providing full support for encrypted web communications (i.e., HTTPS) using DoD PKI server certificates, and protecting the web server and Application from common hacker attacks such as IP spoofing and denial of service. The Application must also provide local auditing and logging of access attempts and invocations of user rights.

Security certification and accreditation of the Application as a stand-alone service (following all pertinent Navy guidelines and policies) remains a responsibility of the Application developer. Applications that are not accredited will not be allowed to take part in the TFWeb architecture.

Refer to Appendix A: DoD Authority References for a complete list of Navy and DoD security policies to be followed.

Legacy Application Authentication Options

To implement SSO, the SSO server relies on encrypted temporary session cookies to securely pass authenticated user IDs between web applications. SSO also provides a means to pass authenticated user IDs through the HTTP request header fields. However, TFWeb recommends that Application developers do not use this data as a means to automatically authenticate users to their Application. Because this data may possibly be spoofed, it should not be trusted as an authentication means, but may be used to provision “familiarity” services, such as a personalized splash page, etc. This leaves Application owners with a few options to authenticate users to the Application, as described below.

During the TFWeb Pilot timeframe, the Application owner may choose to re-authenticate users accessing their Application through the Portal. As this solution does not lead to the goal of end-to-end SSO for the Portal, the Application owner will be asked to migrate to the fully integrated SSO solution once it is determined. Application developers may be required to change the interface through which users authenticate, even if the backend authentication process remains the same.

In-Line HTML Form: The Application should present the user with an HTML form requesting credentials. This form should be designed in-line with the rest of the Application’s interface, and should not be a new pop-up browser window. Level 3 integrated Applications are required to implement HTTP BASIC authentication over SSL, or better.

Dialog Boxes/Child Browser Windows: This is the less preferred, but acceptable authentication interface. The Application may present the user with a system-generated dialog box requesting credentials. However, this dialog box MUST uniquely identify the Application that is asking for credentials to differentiate it from other dialog boxes in use.

Implement SSO Web Server Agent: Application owners may choose to implement a SSO web server agent on the legacy Application’s web server. Using the SSO web server agent will tightly integrate the Application with the Portal SSO solution, which is the long-term goal. There are no licensing issues related to using the SSO web server agent, but there are some technical limitations.

Regardless of the authentication solution chosen for the Application, Application owners are still required to provide a list of all users authorized to access the Application to allow proper operation of the enterprise SSO solution. It must also be noted that any change to the Application’s authentication process should result in a re-evaluation of the Application’s security posture by the System Security Approval Authority (SSAA).

## 6.5 TFW IA Conclusion

The attempt to build a single portal and to web-enable the Navy is a huge undertaking. The IA efforts by both the SPAWAR and ISF engineers are crucial to the successful implementation of this project on a timely basis. Therefore it is imperative that IA be included in all phases of portal development and testing, on both teams , to ensure that security is never an afterthought. The TFW effort is the single unified web portal for both NMCI and IT-21, and therefore it needs to be secure enough to meet needs of both communities. To be truly effective, all personnel associated with operation including developers, web architects, managers as well as staff officers, must understand what the Navy is trying to do in a WEN, and how TFW fits into that overall picture. That is the point, at which the entire security ramifications of the portal are understood, and IA becomes crucial to the overall success of the project.

# Reference Section

All portal implementations for these critical elements must comply with these fundamental Information Assurance policies and practices documents.

***Department of Defense Web Administration Policies and Procedures*, 25 Nov 1998, published under ODSD memo (Hamre) 7 Dec 1998**

**National Policy Governing the Acquisition of Information Assurance (IA) and IA-Enabled Information technology Products,” published by the National Security Telecommunications and Information Systems Security (NSTISSC No. 11), March 1, 2000, available from http://www.nstissc.gov.**

**DoDD 5200.28 *Security Requirements for Automated Information Systems (AISs)* 21 March 1988.**

***Navy-Marine Corps NIPRNET Firewall Configuration Baseline*, 01 Feb 2001.**

***Navy-Marine Corp NIPRNET Firewall Policy*, CNO R 011646Z, Feb 2001.**

***Navy-Marine Corps NIPRNET Firewall Policy Addendum*, CNO R 021933Z, Feb 2001.**

**DoDD 5200.39; *Security, Intelligence, and Counterintelligence Support to Acquisition Program Protection*; 10 September 1997**

**DoDD 5200.40*; DoD Information Technology Security Certification and Accreditation Process (DITSCAP)*; 30 December 1997**

**NSTISSP 200; *National Policy on Controlled Access Protection*; 15 July 87**

**Office of Management and Budget Circular No. A-130, "Management of Federal Information Resources," February 8, 1996**

**SECNAVINST 5239.3; *Department of the Navy Information Systems Security (INFOSEC) Program*; 14 July 1995**

**Chief of Naval Operations (N6) Message NAVADMIN 110/00, *Navy Public Key Infrastructure (PKI) Implementation*, 011504Z MAY 00**

***Information Assurance Technical Framework* (IATF), Version 3, Sept. 2000**

***Web Server Protection* Profile, National Security Agency, draft January 2000. This profile specifies the minimum-security requirements for a web server used in environments where the web server hosts information that must be restricted from public access. As such, information access from the server must be protected from disclosure, must have sufficiently strong mechanisms for access control by web users. Malicious web users must be prevented from modifying or deleting content.**

***Web Browser Protection* Profile, National Security Agency, draft January 2000. This Protection Profile specifies the minimal security requirements for a web browser used in environments where access to information in the host system (including the browser itself) and to the content of web pages must be controlled. The host system must be protected from compromise because of the use of the web browser, either by the transferring of host information to unauthorized users, or by making unauthorized changes to host processes or configuration. Access to web page information must be constrained to the web server from which the page was loaded.**

The portal must comply with these relevant guidance documents:

**National Security Agency, *Guide to Securing Microsoft Windows NT Networks*, Report Number: C4-001R-00, February 3, 2000**

**Trusted Systems Services, *Windows NT Security Guidelines a study for NSA Research* which is available at: http://www.trustedsystems.com/NSAGuide.htm**

**National Security Agency, Router Configuration Guide DRAFT, 2001.**

**National Security Agency, *Guide to Using DoD PKI Certificates in Outlook 2000* DRAFT, 2001.**

**National Security Agency, *Guide to Windows 2000 Kerberos Settings* DRAFT, 2001.**

**National Security Agency, *Guide to Securing Microsoft Windows 2000* DRAFT, 2001.**

**National Security Agency, *Terminal Services* DRAFT, 2001.**

**National Security Agency, *Guide to Windows 2000 Schema* DRAFT, 2001.**

**CSPP - *Guidance for COTS Security Protection Profiles,* Version 1.0, NISTIR 6462, January 2000, available at http://csrc.nist.gov/cc/pp/pplist.htm**

**Common Criteria Project, *Common Criteria for Information Technology Security Evaluation,* Version 2.1., 1999. Available from either: http://csrc.nist.gov/cc/ccv20/ccv2list.htm#DOWNLOAD or http://www.radium.ncsc.mil/tpep/library/ccitse/ccitse.html. Equivalent to ISO FDIS 15408, Parts 1-2-3 (SC27 N2161-2-3).**

**NSA Certified Protection Profiles, available at http://www.radium.ncsc.mil/tpep/library/protection_profiles/index.html**

The portal must comply with these commercial standards:

**Netscape Secure Sockets Layer (SSL) ver 3, Transport Layer Security (TLS, IETF RFC -2246 ), or Secure Hypertext Transfer Protocol (S-HTTP, IETF RFC -2660) requiring dual certification exchange access control.**

**ISO 8879, Information Processing Systems – Text and Office Systems – Standard Generalized Markup Language and World Wide Web Consortium standard XHTML$^{Ô}$ 1.0, "The Extensible Hypertext Markup Language," which is a reformulation of HTML 4 in XML 1.0, Jan 2000.**

**Where applicable, use Signed Document Markup Language (SDML) in MSS web-based apps matching the W$^{3}$C SDML business model targets, using DoD PKI X.509v3 certificates.**

Currently, cryptographic APIs are only used for encryption of unclassified information. But applications using these functions are deployed on unclassified and classified DOD networks, such as Medium Grade Messaging and private web servers. These applications use CAPI to provide the important e-mail digital signature functions mandated by DOD PKI policy.

**Cryptographic APIs must be certified as FIPS 140-1, level 1, compliant.**

The current Mobile Code policy are expanded below:

**Prohibit the use of category 1 mobile code technologies.**

**Use of Java category 2 mobile code will include the COTS security model for (1) Sun Java$^{Ô}$ 2.0 (Security Code Guidelines Feb 2000) or (2) Microsoft J++ (Trust-Based Security for Java April 2000). All Java applets will be signed using Javakey, Signkey, or Authenticode technologies.**

**Scripting languages will comply with EMCA-262/ISO-16262 standard scripting language or Netscape JavaScript version 1.5.**

**Scripting services will comply with World Wide Web Consortium standard XHTML$^{Ô}$ 1.0, "The Extensible Hypertext Markup Language," which is a reformulation of HTML 4 in XML 1.0, January 2000.**

# 7.0 Development Tools and Resources

This section seeks to summarize the myriad of development tools, references and sites that are simply too numerous to include in developer guide. These are the basic tools and resources we have found that may help a developer in the process of creating NAVY Enterprise friendly and compliant applications. Details omitted in the guide may be found here…

| **Tool / Resource** | **Description** | **URL** |
|---|---|---|
| W3C HTML Validator | Checks html code for W3C standards compliance | http://validator.w3.org/ |
| W3C CSS Validator | Checks CSS code for W3C standards compliance | http://jigsaw.w3.org/css-validator/ |
| W3C Link Checker | Customizable web tool to check for broken links etc | http://validator.w3.org/checklink |
| MS Developer Network | Microsoft's developer support web site for MS Technologies. | http://msdn.microsoft.com |
| MSDN Win2k Certification | Desktop & Server App certification & test tool resources | http://msdn.microsoft.com/certification/download.asp |
| UDDI | Universal Discription Discovery and Integration | http://www.uddi.org |
| MS UDDI | Microsoft Interfaces to UDDI and | http://uddi.microsoft.com |

| | | |
|---|---|---|
| | SDK etc. | |
| WSDL | Intro to Web Services Description Language | http://www.learnxmlws.com/tutors/wsdl/wsdl.aspx |
| ISF TOOLS DB | Legacy Apps certification status | https://usplswebh0ab.plano.webhost.eds.net/isftool/Login.jsp |
| EDS ISF Website | Official site of CLINS, tools and resources for NMCI | http://eds.com/nmci |
| DoN CIO Website | Information Management, IT web, resources | http://www.don-imit.navy.mil/ |
| DoN Smartcard Office | Guidence on use of DoN / DOD CAC cards | http://www.donsmartcard.com/Links.asp |
| Source Forge | Open source repository | http://sourceforge.net/ |
| Webopedia | Online Technology Encyclopedia | http://www.webopedia.com |

More to follow…

# Appendix A Checklist

| **Web Enablement** | **Yes** | **No** | **N/A** |
|---|---|---|---|
| Complete Review of Navy Web Enablement documents | □ | □ | □ |
| Complete review of Navy Web Enablement Standards | □ | □ | □ |
| Decompression of application into Web Service | □ | □ | □ |
| Service Module Creation Standards met | □ | □ | □ |
| Is the web server file directory listing disabled? | □ | □ | □ |
| Portal Integration Standards Met | □ | □ | □ |
| Module Server Package for TFWeb | □ | □ | □ |
| Documentation of application data structures and data interfaces | □ | □ | □ |
| Configuration of local application servers or remote module servers | □ | □ | □ |
| Registration Package for TFWeb | □ | □ | □ |
| Migration Package for TFWeb | □ | □ | □ |
| Test Plan for TFWeb Compliance | □ | □ | □ |

| **NMCI Certification and Accreditation Phase** | **Yes** | **No** | **N/A** |
|---|---|---|---|
| Software License and Version collected | □ | □ | □ |
| RFS Completed | □ | □ | □ |
| Media information collected | □ | □ | □ |
| Certification Phase Engineering Review Questionnaire | □ | □ | □ |
| Participation in PIAB Testing | □ | □ | □ |
| NMCI A&RMP, System Level ERQ | □ | □ | □ |
| SSAA Completion | □ | □ | □ |
| IATO Or ATO | □ | □ | □ |

| | | | | | | |
|---|---|---|---|---|---|---|
| DOD PKI Certification for SSL installe | □ | □ | □ | □ | □ | □ |
| Completion of Pop-In-The-Box Process | □ | □ | □ | | | |
| 128-bit SSL encryption is used when traversing any firewall. | □ | □ | □ | | | |
| Reviewed and met all DoN Security Policies and Procedures | □ | □ | □ | | | |
| Reviewed and met all DoN Firewall Policies and Procedures | □ | □ | □ | | | |

# Appendix B Taxonomy

## Enterprise Portal Taxonomy

Navy web enablement is the implementation of interoperable web technologies across the Naval infrastructure allowing subscribers and publishers (users and providers) of content to pull or push services as required to perform operational or business transactions.  A Navy web transaction is the execution of a web-service.  The service centric access for the Navy is depicted in Figure 5-1.

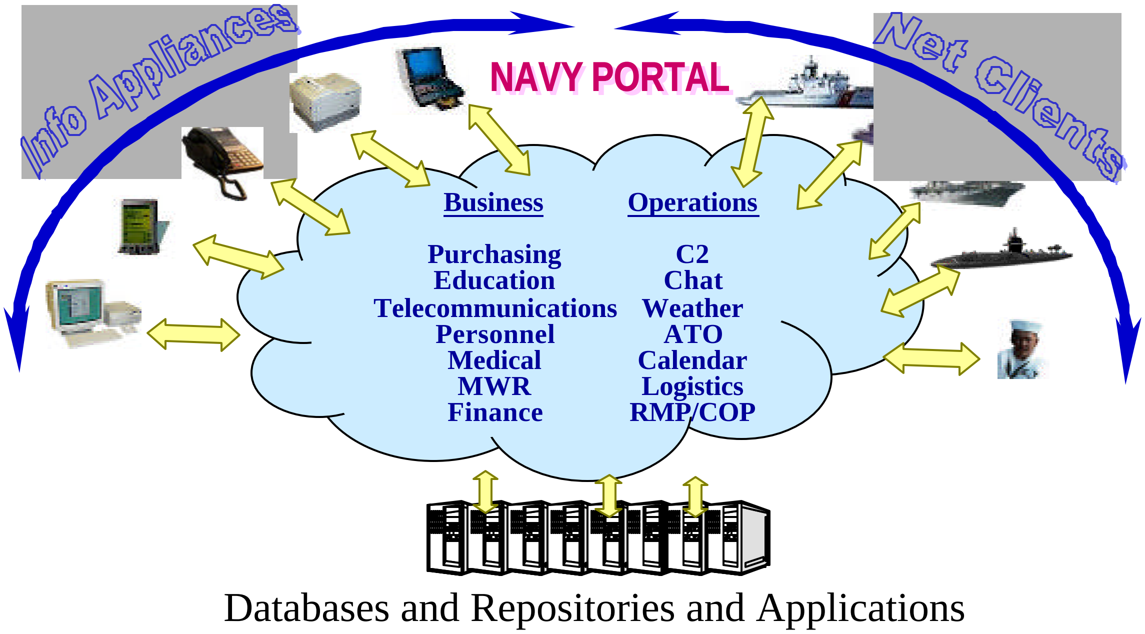


**Figure 5-1:  Service-Centric Access**

This section will discuss guidance and structure for the TFWeb  Portal taxonomy.  This will provide developers with the appropriate background to plan their application migration efforts.

At the highest level, the taxonomy represents the basic set of categories for Navy information sources.  The information involved may be core to the function being performed, or to some other functional area.  The TFWeb  Portal System facilitates the sharing of information between commands and across functional areas.  The Department of Navy Chief Information Officer (DoN CIO) has identified broad information content categories that reside within the enterprise.   Table 5-2 identifies the initial, high-level set of categories.

**Table 5-2:  Initial TFWeb Portal Taxonomy**

| Functional/Resource Area | Program/Resource Sponsor |
|---|---|
| Acquisition | SECNAV RDA/ MARCORSYSCOM |
| Finance | SECNAV FM&C/ HQMC P&R |
| Civilian Personnel | SECNAV CP/ HQMC AR |
| Administration | OPNAV N09B/ HQMC AR |

| **Functional/Resource Area** | **Program/Resource Sponsor** |
|---|---|
| Manpower and Personnel | OPNAV N1/ HQMC MR&A |
| Intelligence and Cryptology | OPNAV N2/ HQMC I |
| Logistics | OPNAV N4/ HQMC I&L |
| Readiness | OPNAV N4/ HQMC PP&O |
| Command, Control and Communications | OPNAV N6/ HQMC C4 |
| Information Warfare | OPNAV N6/ HQMC PP&O |
| Allies | OPNAV N6/ HQMC PP&O |
| Modeling and Simulation | OPNAV N6/ MCSC SE&I |
| Weapons | OPNAV N7/ MARCORSYSCOM |
| Training | OPNAV N7/ TECOM |
| Resources, Requirements, and Assessments | OPNAV N8/ HQMC P&R |
| Scientific and Technical | OPNAV N091/MCCDC |
| Test and Evaluation | OPNAV N091/ MCOTEA |
| Medical | OPNAV N093 |
| Naval Reserve | OPNAV N095 |
| Meteorology, Oceanography, MC&G | OPNAV N096 |
| Religious Ministries | OPNAV N097 |
| Naval Nuclear Propulsion | OPNAV N00N |

[i] ~~Information Management & Information Technology Strategic Plan, FY 2000-2001, Department of Navy Chief Information Officer, page 4.~~

[ii] ~~Ibid, page 12.~~

[iii] ~~This definition of Web Services is adapted from *The Web services (r)evolution, Part 1* by Graham Glass, published in IBM developerWorks XML Zone, November 2000.~~

[iv] ~~Graham Glass, "The Web Services (r)Evolution, Part 1", November 2000, http://www-106.ibm.com/developerworks/webservices/library/ws-peer1.html~~