Navy Enterprise Application Development Guide

Version 2.0 Draft

June 2, 2003

THIS "Navy Enterprise Application Development Guide" IS PUBLISHED FOR INFORMATIONAL PURPOSES ONLY TO ILLUSTRATE APPLICATION PROCESSES AND INTERACTIONS. THE CONTENT OF THIS DOCUMENT SHALL NOT BE CONSIDERED CONTRACTUALLY BINDING. ALL ISSUES ASSOCIATED WITH THE NMCI CONTRACT N00024-00-D-6000 SHALL BE REFERRED TO THE PROCURING CONTRACTING OFFICER AT 619-524-7388.

Change History

The following Change History log contains a record of changes made to this document.

Entries should be made in descending order, with most recent changes at the top of table.

Published / Revised Date	Version # [0.XX for unpublished documents]	Change Author(s)	Section / Nature of Change
June 02, 2003	2.0	Mark Zabriskie, Temo Villanueva, Steve Parker, Dan Tarkenton	V2.0 available in online presentation via TFWeb site. Various TFWeb section modifications Updated (0.2) Expand scope to align w/ other guides... (LATG,NRDDG, LSTG) (appendix B)Updated POC's (2.1.6.3) (updated DONXML WG reference) (2.1.11.1.24) extended CLIN 029 to web application hosting paragraph. (3.1.1) added reference to new Navy Marine Corps Portal (NMCP) Policy Guidance Memorandum
February 10, 2003	1.12	Mark Zabriskie, Bill Adsit, Taylor Holmes, Bette Fondas, Joel Brown	Updated 2.1 Navy Enterprise Portal Integration and TFW, Updated 3.1 NEP Integration Processes and TFWeb, updated APPENDIX E: PRI Data, updated APPENDIX F: Portal CSS, added header text to APPENDIX I:, updated APPENDIX J: Case Study 1: Employee/Member Self Service, Updated APPENDIX M: NEP Developer's Integration Checklist, Added APPENDIX N: Navy Mobile Code , Global replacement of www.homeport.navy.mil with portal.tfw.navy.mil, various additional typographic and clarification corrections.
January 09, 2003	1.11	Steve Parker	Updated links, logos, and POCs.
October 29, 2002	1.1	Temo Villanueva	Appendix I FAM chart updated.
October 25, 2002	1.0	Mr. Steven Ehrler (PEOIT) / Ms. Monica Shephard (Dir TFWeb)	PEOIT and TFWeb Directors added
September 26, 2002	0.10	Final draft by NEADG review team	Final Draft Review
August 23, 2002	0.09	NEADG internal review team	Pre Draft Review
August 16, 2002	0.08	NEADG internal review team	Incorporates contributions from subject matter experts and bin leaders post 9-10 July TFWeb developer conference

Published / Revised Date	Version # [0.XX for unpublished documents]	Change Author(s)	Section / Nature of Change
June 29, 2002	0.07	James Gallagher, Jackie Montgomery	Final administrative update to entire guide
June 26, 2002	0.06	John Stafford, Temo Villanueva, Carl Prantl	Section 1.4 reword, replaced sect 2.1, & Appendix M, reword, 4.0 Introduction, 4.1.6 reorg, 4.1.4 & 4.1.7 replace Taxonomy & Service meta data sections, Glossary TFWeb term update, added TFWeb Checklist to Appendix M, cross-referenced Table 1 to Appendix L
June 25, 2002	0.05	Capt. Copelof/Temo Villanueva	Further administrative and conceptual cleanup
June 25, 2002	0.04	Carl Prantl John Stafford Temo Villanueva	TFWeb Process Section 4.1 updates and misc. updates/changes/mods
June 24, 2002	0.03	Several Authors	Administrative, content, and reorganization changes made, listed in "comments," spreadsheet available upon request; addressed several comments from initial review team; case studies added.
June 19, 2002	0.02	Internal Comments Review	Select comments were addressed from various in-house participants.
June 14, 2002	0.01	Temo Villanueva, PEOIT villanuevat@saic.com Michael Donovan, ISF michael.donovan@eds.com Carl Prantl, TFWeb/Mitre cprantl@mitre.org L. Rajeev Parekh, TFWeb/N6 parekhrv@clf.navy.mil Taylor Holmes, TFWeb/SSC holmest@spawar.navy.mil Geary Sutterfield, TFWeb glsutter@mitre.org Mark Zabriskie TFWeb/SPAWAR-SD zabriski@spawar.navy.mil	First TFWeb publication restructure intended for circulation and comment among CDA developers

DOCUMENT PROPERTIES

Filename: NEADG_v2.0_DRAFT.doc

Owner: PEOIT (http://www.peo-it.navy.mil)/TFWeb (http://www.tfw.navy.mil).

The following is the format for feedback on this version of the NEADG:

Does this guide give you what you need?

Does this guide overly constrain you and how?

Please send feedback to

Steve			Parker
PEO(IT)	Developer	Guidance	Support
parkers@saic.com			
703-868-8334			

Temo			Villanueva
PEO(IT)	Developer	Guidance	Support
villanuevat@saic.com			
858-826-5168			

For additional information, please register your comments with the TFWeb Open Source Site at

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

Credits:

This document was the collective efforts of many organizations throughout industry, DoD, and the Navy. In particular, the following organizations are recognized for their respective contributions:

PEOIT for NMCI
NAVAIR for TFWeb
DON CIO
MITRE
NAVFAC
NAVFACENGCOM HQ
NAVSEA
SAIC
SPAWAR
SPAWAR Systems Center San Diego
SPAWAR Systems Center Charleston
TFWeb AMCS Team, Norfolk
TFWeb AMTS Team
TFWeb Development Team
VARIOUS DON CDA CONTRIBUTORS

TABLE OF CONTENTS

Figure Page viii
Table Page ix

0. INTRODUCTION 11
0.1 Purpose 11
0.2 Document Scope 12
0.3 Approach 14
0.4 Development Scenarios 15
0.5 Intended Audience 17
0.6 Base Level Information Infrastructure (BLII) 18
0.7 Information Technology for the 21st Century (IT21) 19

1. PREPARATION AND ANALYSIS – PHASE I 20
1.1 Web Application Analysis 20
1.1.1 Requirement to Web Enable 20
1.1.2 Determine Enterprise Portal Integration Goals 20
1.1.3 Functional Area Manager Rationalization 21
1.1.4 Supportability and Maintainability 22
1.1.5 Factors Affecting Web Enablement 23
1.2 NMCI Desktop Application Analysis 24
1.2.1 Legacy Applications Transition Guide (LATG) 24
1.2.2 Navy Application Rationalization 24
1.2.3 Information Strike Force Tools Registration 24
1.2.4 ISF Tools Database Description 25

2. DESIGN AND DEVELOPMENT – PHASE II 27
2.1 Navy Enterprise Portal Integration and TFWeb 27
2.1.1 NEP Scope 27
2.1.2 Assumptions 27
2.1.3 Target Audience 27
2.1.4 Prerequisite Knowledge 28
2.1.5 Getting Started with the Navy Enterprise Portal 29
2.1.6 Portal Content Integration 38
2.1.7 Portlet Interface 45
2.1.8 User Facing Service Interface 52
2.1.9 Portal Services Interfaces 56
2.1.10 Infrastructure and Design Considerations 67
2.1.11 Development Resources 71
2.2 NMCI Integration 72
2.2.1 Boundary/Network Interface Specifications 72

3. CERTIFICATION AND DEPLOYMENT – PHASE III 78
3.1 NEP Integration Processes and TFWeb 78
3.1.1 NEP Service Certification Process 78

3.1.2 Information Assurance Certification and Accreditation (IA C&A) 80
3.1.3 Service Migration Package Submittal 82
3.1.4 NEP Taxonomy 82
3.1.5 NEP Service Rationalization 84
3.1.6 NEP Service Migration Package 85
3.1.7 Service Registration Metadata 85
3.1.8 Service Lifecycle Management 89
3.1.9 TFWeb Test Processes 89
3.1.10 NEP IT Governance 90
3.1.11 NEP Service Deployment 90
3.2 NMCI Integration Process and ISF 91
3.2.2 Before Visiting NMCI for an Engineering Review 93
3.2.3 Certification Lab Activity 94
3.2.4 Certification Lab Process 95
3.2.5 Before Deployment/Migration 98
3.2.6 Deployment/Migration 101
4. REFRESH AND RETIREMENT – PHASE IV 103
4.1 System Changes 103
4.1.1 Emergency Production Fixes 103
4.1.2 Recertification Procedures 103
4.2 System Retirement/Sunset 104
APPENDIX A: GLOSSARY OF TERMS AND ACRONYMS 105
APPENDIX B: REFERENCES 113
APPENDIX C: FREQUENTLY ASKED QUESTIONS 116
APPENDIX D: POINTS OF CONTACT 118
APPENDIX E: PRI DATA 120
APPENDIX F: PORTAL CSS 128
APPENDIX G: APPLICATION SECURITY 140
APPENDIX H: URL REWRITE GUIDELINES 146
APPENDIX I: FUNCTIONAL AREA MANAGEMENT (FAM) SECNAV REFERENCE 150
APPENDIX J: CASE STUDY 1: EMPLOYEE/MEMBER SELF SERVICE INTEGRATION 153
APPENDIX K: CASE STUDY 2: NTIRA WEB PROJECT CASE STUDY (PHASE I) 158
APPENDIX L: NMCI APPLICATION RULE SET 165
APPENDIX M: NEP DEVELOPER'S INTEGRATION CHECKLIST 167
APPENDIX N: NAVY MOBILE CODE POLICY 169

LIST OF FIGURES

Figure **Page**

Figure 1: Navy Enterprise and NMCI Process Guides 14
Figure 2: Phases 15
Figure 3: Tabbed Workplaces, User Profile Access, and InfoStore Components 30
Figure 4: InfoStore Contents (Channels and Library) 30
Figure 5: Portlet Examples 31
Figure 6: Three-Tier Architecture and Portal 32
Figure 7: Navy Enterprise Portal 33
Figure 8: Support for Multiple Clients and Dynamic Content Rendering 34
Figure 9: NEP Execution Sequence Diagram 35
Figure 10: Good and Bad Portlet Examples 40
Figure 11: Example of a Portlet Using Reference Integration 41
Figure 12: Example of a Portlet Using External Content Integration 42
Figure 13: Example of a Style Sheet Changing 43
Figure 14: Portlet Interface Section Scope 45
Figure 15: Portal Precondition Processing 48
Figure 16: Portal Response Processing 52
Figure 17: User Facing Service Interface Section Scope 53
Figure 18: Typical UFS Activities 55
Figure 19: Portal Services Interface Section Scope 57
Figure 20: Network Boundaries 73
Figure 21: NEP Service Certification Process 80
Figure 22: Overview of DITSCAP Phases * 81
Figure 23: Application Integration & Testing Lab (AIT) Process 96
Figure 24: Sample Screen 1 with Style Tags 133
Figure 25: Sample Screen 2 with Style Tags 134
Figure 26: Sample Screen 3 with Style Tags 135
Figure 27: Sample Screen 4 with Style Tags 136
Figure 28: Sample Screen 5 with Style Tags 137
Figure 29: Scope of Application Security Appendix 140

Figure 31: Portal Friendly Reference Integration Portlet 156
Figure 32: Reference Integration Portlet Utilizing Portal CSS 157
Figure 33: Fiscal Reporter UFS – Fiscal Data View 158
Figure 34: Fiscal Reporter UFS – Fiscal Graph View 159
Figure 35: Battle Group Mission Status UFS 160
Figure 36: Mission Status by Revision UFS 161
Figure 37: Mission Status by Platform UFS 162
Figure 38: NTIRA Software Components 164

LIST OF TABLES

Table **Page**

Table 1: Web Application Development Scenarios 16
Table 2: Legacy Application Development Scenarios (candidates for web enablement) 16
Table 3: Required Implementation Items by Integration Type 38
Table 4: Portlet Content Integration Characteristics 44
Table 5: Portlet Service Request 49
Table 6: Portal Response Processing 51
Table 7: Portal Services 58
Table 8: URL Rewrite Request 59
Table 9: URL Rewrite Response 60
Table 10: URL Rewrite Postconditions 60
Table 11: WSEE Request 61
Table 12: WSEE Response 64
Table 13: WSEE Postconditions 64
Table 14: XML Transformation Request 66
Table 15: XML Transformation Response 67
Table 16: XML Transformation Postcondition 67
Table 17: PKI References 70
Table 18: Service Metadata Information 86
Table 19: Service Contact Information Description 89
Table 20: NMCI Help Desk Recommended Developer Fields 100

Table 21: Useful Hyperlinks (all external links) 113

Table 22: NMCI Transition Teams 113

Table 23: Related Navy Messages..... 114

Table 24: Important Standards (all external links) 114

Table 25: NEADG POCs..... 118

Table 26: TFWeb AMCS POCs 118

Table 27: PRI Data Element Definition..... 120

Table 28: SS Tag Descriptions for Style Sheets 137

Table 29: Security Implementation Combinations 142

Table 30: Example 1: Comparison 142

Table 31: Example 2: Comparison 143

Table 32: Example 3: Comparison 143

Table 33: Example 4: Comparison 144

Table 34: Example 5: Comparison 144

SUPPLEMENTS

A. NMCI Release Development & Deployment Guide (NRDDG)

The Navy Program Management Office (PMO), in concert with the Information Strike Force, has developed the NMCI NRDDG to provide detailed information and guidance to developers interested in migrating content, introducing new applications, or changing existing applications within NMCI. The guide is a consolidated source of information, guidance, and direction to developers who build or modify applications and/or to the acquirers of applications intended for use specifically within NMCI. The NRDDG is available for download at the following URL: http://www.nmci-eds.com/transition.htm#Release .

0. INTRODUCTION

The Navy Enterprise Application Development Guidance (NEADG) project originated from a collaborative effort between the Program Executive Office for Information Technology (PEO-IT) and the Task Force Web (TFWeb) office to combine the Navy Marine Corps Intranet (NMCI) Application Resource Guide (ARG) document with the Task Force Web Integrated Developer's Guide (TFWeb IDG) document. This effort was established to raise developer awareness of the different policies, processes and procedures governing their software releases on two of the newest infrastructures in the Department of the Navy (DON), NMCI and the Navy Enterprise Portal (NEP). The NMCI ARG was designed from an ashore, desktop-centric perspective to describe the NMCI operating environment in which new, emerging, and legacy applications and systems must operate, while the TFWeb IDG provided detailed guidance on developing n-tier web-based or web-enabled applications and services, as well as modifying existing applications for seamless integration into the NEP.

The initial NEADG effort was successful in combining these two guidance projects. Now in its latest iteration, it intends to incorporate and/or link to additional guidance projects related to other DON infrastructures such as Information Technology for the 21st Century (IT21), and Base Level Information Infrastructure (BLII). Additionally, a web-based approach is being leveraged to enhance usability and maintainability as well as to enable more seamless integration of related guidance.

0.1 Purpose

This guide provides the integrated overarching technical direction for enterprise application development within, or transition into, the DON Enterprise IT Environment. Any application that must integrate with DON enterprise services or platforms should be built and maintained in accordance with the standards, policies, and processes within this guide.

This guide addresses the following DON platforms and services:

NEP

IT-21

BLII

It provides guidance on how to handle legacy applications and how to build new applications across these environments. Specific emphasis is on migrating applications to a web-based environment that conforms to the NEP standards.

As the NEP is fully deployed across these environments (IT-21, NMCI, and BLII), it provides seamless information exchange and will transform the nature of application development and data management across the Navy. This guide provides an overview of these diverse environments, but concentrates on the standards and processes needed by developers to deploy enterprise applications that rely on single authoritative data sources and that are accessible by users from anywhere within the enterprise. The guide provides information that will assist developers in creating migration plans for legacy applications that will migrate into the NEP and thus meet the NMCI, IT-21, or BLII requirements. It also provides guidance so all new application development will be fully compliant with the NEP standards.

The guide concentrates on the NEP and NMCI processes and wherever possible shows where these processes have been aligned. As both the NEP and NMCI processes are continuing to evolve, the alignment of these processes will also continue to be refined. As the BLII is deployed, the integration of these processes will also be included in later editions of the guide.

0.2 Document Scope

The scope of the NEADG is limited and makes assumptions about the target audience and the level of knowledge within the developer community. Wherever possible, this guide is based on commercial standards and builds on the standards and instructions already promulgated for the Navy that are listed in Appendix B. Readers are assumed to have a firm working knowledge of these standards and references. In addition, web sites that provide detailed information about the technologies, standards, interfaces, and protocols used are provided in the list of references and in Appendix B. Additional information, including frequently asked questions, detailed coding examples, and access to technical support, is available via the TFWeb Open Source Site at https://tfw-opensource.spawar.navy.mil. NMCI-related content is available at http://www.nmci-isf.com.

The scope of this guide is limited to "Navy Enterprise" application developers, typically central design authorities (CDAs), for developing, configuring and migrating applications that comply with NEP, NMCI, IT-21, and BLII standards.

This document details the overall design of the NEP and NMCI applications integration infrastructure and describes technical requirements as well as logical and physical architecture. Architecture information is presented by subcomponent and includes conceptual issues and items relating to policy/security, management functionality, and deployment.

This document is built upon the DON XML Vision (promulgated in March 2002), which contains the foundation on which to exploit XML technology by identifying "best fit" applications of this technology in DON applications and architectures. All use of XML by developers should be in consonance with the guidelines of the DON XML Vision and subsequent DON XML standards (see APPENDIX B: REFERENCES).

Those interested in NMCI specific guidance may find other guides of particular interest, including the Legacy Applications Transition Guide (LATG), the Legacy Systems Transition Guide (LSTG) and the NMCI NRDDG (a supplement to the NEADG). These guides focus on the configuration of applications and systems targeted for NMCI. They are briefly described below.

LEGACY APPLICATION TRANSITION GUIDE (LATG)

The LATG presents a complete baseline overview of the Legacy Applications Rapid Certification Phase of the Legacy Applications Transition Process. It is for Navy Marine Corps Intranet (NMCI) customers involved with transition activities. Information Strike Force (ISF) and Government program management personnel worked in close cooperation to design the processes, procedures, and policies described in the LATG. The Guide describes in detail the roles and responsibilities of those organizations and positions involved in transitioning Legacy Applications through the Rapid Certification Phase.

NAVY APPLICATION DEVELOPMENT GUIDANCE (NEADG)

The purpose of the NEADG is to provide detailed information and direction to developers tasked with migrating applications, content, and services into the DON Enterprise IT Environment. This effort seeks to align current software development community practices within the Navy and Marine Corps with the best practice vision embraced by the DON and DOD of web enabled enterprise applications in a way that is extensible, scalable, and open in its use of current standards and cutting-edge

technologies. Additionally, the NEADG seeks to present guidance information in a way that is easily human and machine searchable and that adds value to the consumer of information by aligning the guidance to their specific activities and roles. This will be accomplished by mapping constituent guide content to searchable Meta data categories that will allow distinct renderings or views of the original data. The web-based platform will enable the collection and packaging of content from other guides focused on the overall purpose of guiding CDAs in transitioning of applications toward the DON Enterprise IT Environment.

NMCI RELEASE DEVELOPMENT AND DEPLOYMENT GUIDE (NRDDG)

The NMCI NRDDG is a consolidated source of information, guidance and direction to developers and application owners who build and/or modify applications as well as the acquirers of applications intended for use within NMCI. As a supplement to the NEADG, the NRDDG was written to support the CDA in the development and deployment of releases that will operate within NMCI. For web based application guidance, the user should refer to the NEADG, the TFW and NEP.

LEGACY SYSTEMS TRANSITION GUIDE (LSTG)

The LSTG provides both an approach and the associated processes to successfully transition systems from legacy environments to the NMCI environment while maintaining or improving system performance and availability. The LSTG provides the Site Representative, Central Design Authority (CDA), and the Program of Record/Program Manager (POR/PM) with the unique processes, tools/templates, and documentation guidelines to plan and execute the transition of their respective systems to the NMCI environment.

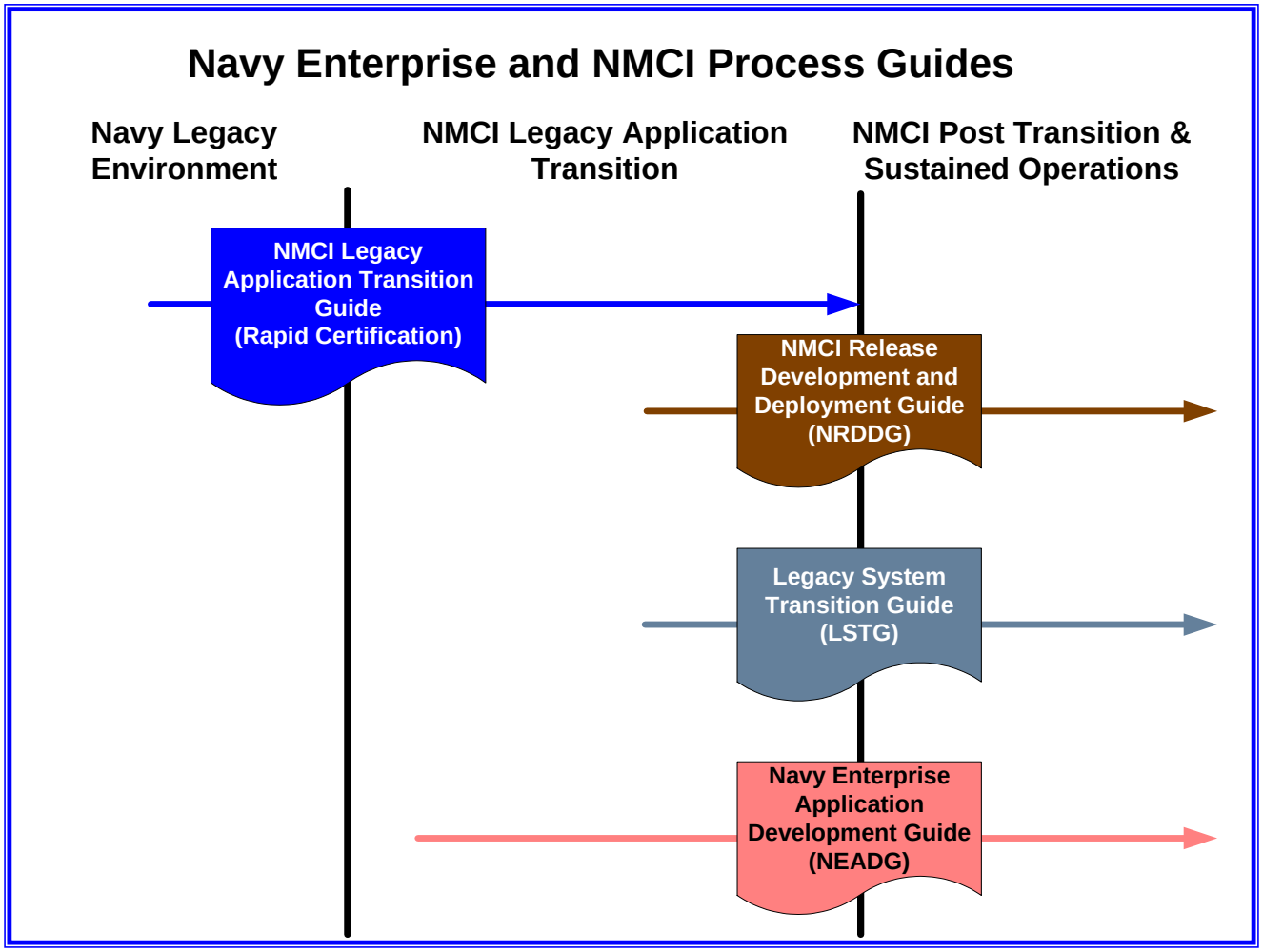


Figure 1: Navy Enterprise and NMCI Process Guides

0.3 Approach

The approach followed by this guide to provide integrated overarching guidance is based on a cooperative effort between several organizations within the DON that are each responsible for producing and maintaining guidance and policies regarding specific platforms and services in the DON. This guide does not attempt to describe the internal details of any of the individual services or platforms. These are documented separately through their respective programs and incorporated or referenced (cross-linked) as appropriate. Instead, this guide seeks to collect all relevant guidance and incorporate it into an overarching construct that enables the integrated delivery and presentation of guidance to the end-users. In addition, this guide attempts to minimize the complexity of the integration process by viewing the enterprise services and platforms and the content/applications to be integrated as “black boxes”. Much of this guide focuses on providing detailed information about the interfaces between the “black boxes” vice describing in great detail the inner workings of the enterprise components. This guide provides information on the interfaces on which application developers need to concentrate.

In order to execute this approach, a web-based platform is used to enhance usability and maintainability as well as to streamline the integration of the guidance. This Web-based Integrated Guidance Platform effectively realizes the NEADG approach in a flexible online environment that provides a two-way information channel connecting the authoritative experts of the various enterprise platforms and services and their respective guidance content to the end-users who must integrate applications with these platforms and services.

This document focuses on the activities of application developers or CDAs and is structured around four phases, as follows:

(Click on any phase to go directly to the related section of the document.)

Figure 2: Phases

This document presents some key considerations for Phase I, including the implementation of thin- and thick-client applications consistent with both the NMCI desktop environment and the NEP. This document does not address in detail the processes of rationalization and portfolio management that should be used to augment the activities described in Section 4. The majority of this document focuses on Phases II and III to enable developers to develop and certify applications to run on the NEP and convert legacy applications to web applications consistent with the NEP. Phase IV is presented in this document to complete the system life cycle.

Based on a review of the Joint Technical Architecture (JTA) standards, this document is determined to be in compliance. Compliance at lower levels of the network open system interconnection (OSI) model is assumed based on compliance statements in related architecture documents.

0.4 Development Scenarios

In the DON, application developers may encounter the following high-level scenarios with their application development efforts:

Table 1: Web Application Development Scenarios

#	Scenario (application characteristics)	Basic Considerations	Document Sections
1	Web application accessible via a Navy Enterprise Portal instance.	Develop using World Wide Web Consortium (W3C) development standards. Reference this document for applicable NMCI, DON, and DoD standards. Test application and register it with TFWeb. See https://tfw-opensource.spawar.navy.mil and http://www.nmci-isf.com	2.1.5 Getting Started with the Navy Enterprise Portal Table 3: Required Implementation Items by Integration Type) References: Application Security
2	Web application or service integrated into a Navy Enterprise Portal instance	Develop using J2EE, .NET, and/or W3C standards. Refer to this document for DON and other DoD technology. See https://tfw-opensource.spawar.navy.mil and http://www.nmci-isf.com.	Table 3: Required Implementation Items by Integration Type) 2.1.6.3 Content Integration Application Security

Table 2: Legacy Application Development Scenarios (candidates for web enablement)

#	Scenario (application characteristics)	Basic Considerations	Document Sections
3	New and Emerging stand alone Desktop application slated only for NMCI GFE workstation (32-bit application)	Test application against Windows 2000 application specification. Follow NMCI certification process including Information Strike Force (ISF) Tools Database registration/rationalization. See http://msdn.microsoft.com/certification and http://www.nmci-isf.com. (Legacy Applications Database Link)	1.2 NMCI Desktop Application Analysis 3.2 NMCI Integration Process Application Security **Error! Reference source not found.** Refer to "Supplement A" NMCI Release Development Deployment Guide (NRDDG).

#	Scenario (application characteristics)	Basic Considerations	Document Sections
4	Client/server applications using thick-client technology, slated for NMCI environment only	Test client application against Windows 2000 desktop specification. Test server components against Windows 2000 server specifications. Use NMCI certification process, including ISF Tools Database registration/rationalization. See: http://msdn.microsoft.com/certification and http://www.nmci-isf.com	1.2 NMCI Desktop Application Analysis 3.2 NMCI Integration Process Application Security **Error! Reference source not found.** Refer to Legacy System Transition Guide (LSTG)
5	Terminal services access to legacy network (screen scraper)	Consider integration with Citrix Metaframe thin client multimedia control of application solution available in NMCI v2.0 via NEP. See: https://tfw-opensource.spawar.navy.mil and http://www.nmci-isf.com	2.2.1.8 Terminal Services Application Security NMCI Contract Line Item Numbers (CLINs) http://www.nmci-isf.com/clinlist.htm **Error! Reference source not found.**
6	Terminal emulation application on legacy network	Typically a temporary access solution. Integrate with WRQ Reflections, Reflection Launch, or other solutions available via NEP in NMCI v2.0. See: https://tfw-opensource.spawar.navy.mil and http://www.nmci-isf.com.	NMCI Legacy Data Access Section App G: Application Security Released in NMCI 2.0 **Error! Reference source not found.**

0.5 Intended Audience

While the principal audience of this version of the document is intended to be developers, architects, and engineers or CDAs, managers may find Preparation and Analysis – Phase I of particular use. Developers may find

Design and Development – Phase II to be more useful. Execution/maintenance individuals may find Certification and Deployment – Phase IIIand Refresh and Retirement – Phase IV of particular interest.

Interested audiences may include the following:

- NEP Integration Architects/Engineers
- NMCI Integration Architects/Engineers
- NEP Application Developers
- NMCI Application Developers

- Subject Matter Experts (SMEs) involved in NMCI and web-enabled application development

This document may be expanded in the future to support application and content owners more thoroughly.

0.6 Base Level Information Infrastructure (BLII)

In support of Navy activities outside the continental United States (OCONUS), the Navy utilized the BLII OCONUS modernization project to implement an enterprise-wide, shore IT network capability. This network provides fully integrated and interoperable end-to-end connectivity, as well as secure access to the full range of data services. It ensures the reliability, availability, and integrity of Navy information systems and provides the infrastructure needed to protect, defend, secure, and support the Navy's mission-critical capabilities.

The objective of the BLII OCONUS modernization project was to install an IT infrastructure that is fully interoperable with the CONUS NMCI architecture and concept of operations, yet fulfills OCONUS requirements. The delivery order was awarded to General Dynamics in May 2001. Completion is scheduled for the second quarter fiscal year 2004.

Space and Naval Warfare Systems Command (SPAWAR) deployed the Navy Network OCONUS under the BLII OCONUS modernization project at 16 major fleet concentration areas: Europe: Naples (to include Naval Support Activity, Gaeta), London, Rota, Souda Bay, Sigonella, and La Maddalena; Pacific Far East: Yokosuka, Sasebo, Misawa, Atsugi, Okinawa, Korea, Guam, Singapore, and Diego Garcia; and the Middle East: Bahrain. This Navy Network OCONUS replaced aging Navy IT infrastructure and serves an OCONUS population of approximately 27,000 users. It consists of local area networks (LANs) at priority sites and connects by base area networks (BANs), metropolitan area networks (MANs), or wide area networks (WANs) throughout the region.

Each regional commander in chief (CINC) directs and controls operation of the Navy Network OCONUS through the O&M Lead Command and the regional Information Technology Service Centers (ITSCs). The O&M Lead Command operates the ITSC, the hub for information exchange, in coordination with the Information Technology Outreach Centers (ITOCs). Each ITSC coordinates mutual support with the other ITSCs, the joint CINC theater operations center, Commander Navy Network Operation Command (CNNOC), the regional Defense Information Systems Agency (DISA) operations center, and joint and coalition commands to ensure computer network defense. A Regional Information Technology Council (RITC) has been established in each region to assist in the identification of requirements. The RITC includes representatives from the CINC (chair), regional Naval Computer and Telecommunications Area Master Stations (NCTAMS), Naval Computer and Telecommunications Stations (NCTS), and all Echelon II commands that have claimants in the region.

The mission of the Navy Network OCONUS is to operate an IT network for regional Navy customers. This network supports end-to-end connectivity with a standard configuration in a secure environment for classified and unclassified IT processes. The goal is to support common applications across the OCONUS network with an exit strategy of contractor-provided seat management.

The Navy Network OCONUS is operated and maintained by the respective area Fleet CINCs. Each Fleet CINC, with CNNOC, shall develop a plan and processes that define the management, operation, and maintenance of the network in support of its claimants. This enterprise IT support is based on a service provider relationship with the customers/claimants in each region represented at the RITC and a two-tiered organization of ITSC and subordinate ITOCs.

Pending the initial operational capability of OCONUS BLII, the full requirements for application integration have not been defined. However, the concept of operations and network interoperability processes for BLII are consistent with those of IT-21. Concerning development of applications and services for BLII, those guidelines delineated for IT-21 provide a low risk approach for integration. The actual certification and approval process for BLII applications will be provided as soon as they become available from the BLII Program Office.

0.7 Information Technology for the 21st Century (IT21)

There is an initiative underway to create a common infrastructure for all shipboard networks so that applications that must reside in ashore production environments need target only a single platform and will be interoperable across all shipboard environments. This initiative is IT-21. This guide aims to provide the technical guidance for application developers targeting the ashore environment by appropriately referencing and incorporating the existing guidance for IT-21. The NEADG currently identifies the areas where guidance will be extended to provide more detail regarding IT-21. More formal incorporation is planned for next version.

1. PREPARATION AND ANALYSIS – PHASE I

1.1 WEB APPLICATION ANALYSIS

This section analyzes applications that fall under scenarios 1 and 2 in Table 1: Web Application Development Scenarios), representing web-based, web-enabled, or web-accessible applications. Developers of web applications accessible via the NEP must register with the TFWeb site at https://tfw-opensource.spawar.navy.mil. Updates, migration paths, developer steps, further guidance, and resources are available to developers at this site.

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

The process begins when the service provider determines the applicability of migrating the application to the NEP. Next, a review of existing services and data sources is conducted to identify possible duplication. Once the decision is made to migrate the application to the NEP, the developer will register the application with TFWeb. A member of the Application Migration Customer Support (AMCS) team is assigned to assess the application, identify overlapping applications and data sources, and assist in compiling the migration package for submission. The migration package is reviewed by the AMCS team and is sent to the test labs for analysis prior to integration.

1.1.1 Requirement to Web Enable

Naval Message R 171442Z APR 01 (see Table 23: Related Navy Messages)) requires all enterprise applications to "be accessible via the NEP to provide a single common access point for all Navy application services and information dissemination, allowing Navy enterprise-wide process reengineering and empowering personnel at all command levels."

1.1.2 Determine Enterprise Portal Integration Goals

Most commands and Navy developers currently have some experience with the World Wide Web (WWW). Many commands have a simple web site of static content. Some have started to move interactive information online to gather and distribute data. A few have begun to move the business logic that is traditionally distributed to users via client-server applications to a web environment.

What is a "web-enabled" application? While it clearly goes beyond a static web site, an exact definition is difficult to discover. For the Navy, web enablement is compliance with NEP standards.

These standards require browser independence for cross-platform functionality and compatibility with emerging standards for wireless and nontraditional clients. Web-enabled applications may use any suitable technology, e.g., Java, J2EE, JSP, ASP, .NET, along with many others provided the application complies with vendor-neutral interface standards described in

Design and Development – Phase II and APPENDIX B: References. NEP applications are encouraged to use XML to enable interoperability and enterprise data sharing.

Not all applications require the full range of capabilities provided by the NEP. Some existing web sites may use reference integration to make their content rapidly accessible through the NEP. This limited level of integration is unlikely to be satisfactory as a long-term solution because of the inconsistent interface presented to users. External content integration allows developers to use traditional web development tools in a structured environment. This allows applications to be tiled in a window incorporating multiple applications simultaneously. It also permits use of graphically rich environments and complex user interfaces.

Content integration is the ultimate goal for all NEP applications. In this type of integration, the application uses standard NEP style sheets. These style sheets provide a consistent look and feel for all applications available in the NEP. In addition, the application supports resizable portlet panes. These features of content integration enable a true "portal-friendly" application. See Section 2.1.6 Portal Content Integration for more information on content integration types.

Developers have many different reasons to integrate their applications in the NEP. Some simply desire to meet the requirements of naval regulation. Others view the NEP as a way to raise the profile and the quality of their formerly internal data sources as they are utilized Navy wide. Many look to write applications closely tailored to the needs of the war-fighting consumer while utilizing the shared security and afloat hardware infrastructure the NEP provides.

Regardless of a developer's current web posture, there are key things that a developer, program, application, or content manager should consider before integration in the NEP:

- Functional area management
- Review of existing services and content
- Supportability and maintainability

1.1.3 Functional Area Manager Rationalization

As prescribed in SECNAVINST 5000.36, the designated Functional Area Managers (FAM)s are responsible for the execution of approved Department of the Navy (DoN) business processes (See http://neds.nebt.daps.mil/Directives/5000_36.pdf or APPENDIX I: Functional Area Management (FAM) (Reference for a snapshot of the FAM process). DoN developed the processes to assist FAMs with adjudication decisions for the migration, consolidation or retirement of applications and databases used within their respective functional areas, and for those applications and databases respective FAMs own that have cross-functional impact on Navy functions and systems.

FAMs develop and manage IT application and database investment portfolios to ensure that current technology strategies are aligned with business administrative and war fighting strategies. Additionally, FAMs collaborate with the DoN Chief Information Officer (DoN CIO) and DoN Information Executive Committee Service for Joint Applications to ensure DoN processes and procedures are followed. The FAMs coordinate their actions with the Office of the Secretary of Defense (OSD) principal staff assistants for Joint applications, DFMMP Manager for financial and business systems, Director of Navy Marine Corps Intranet (NMCI), Echelon II and Major USMC Commands, Central Design Activities (CDA), Application Owners, and Programs of Record (POR).

FAMs manage and oversee the activities of assigned Functional Data Managers (FDMs) and Functional Namespace Coordinators (FNCs). FDMs and FNCs are responsible to work with FAM business process owners to produce and monitor the use of data and functional namespaces generated across functional activities and systems.

CDAs should first consult the DoN Application and Database Management System (DADMS) (see https://www.dadms.navy.mil).database for application status before executing any major application action (such as deploying onto the NMCI) that may have impact on the respective FAM or their associated business processes.

1.1.3.1 Market Review of Existing Services and Content

Managers should review existing applications (commercial or otherwise) for overlapping capabilities. It is recommended that representative users be solicited for possible alternative solutions. The ISF Tools Database and DADMS form a comprehensive listing of Navy IT software.

In most cases, there is not total overlap between two applications. In a case of functional overlap, required functionality should be documented by the program manager and validated by the user. TFWeb works with program managers to develop innovative strategies to merge IT development efforts.

1.1.3.2 Registered Services and "Best of Breed" Determination

When the developer registers an application with TFWeb, AMCS verifies that there are no applications in the NEP that provide overlapping functionality or content.

In the event that there is overlapping functionality, AMCS will work with the application owners and the FAM to analyze and document the overlap and develop a migration plan. If a migration plan cannot be agreed upon, a recommendation is made to the TFWeb Executive Steering Group (ESG), which determines which applications are allowed to integrate with the NEP. The ESG will also recommend solutions to OPNAV and Echelon II commands to resolve application/data overlap.

The decision is based upon the following criteria:

- Technical and architectural analysis (including compliance with Joint, DON, and NEP standards)
- Operational Advisory Group (OAG) evaluation of applications to meet operational environmental needs.
- Several functional groups already exist and are utilized when possible.

The OAG will be comprised of members from the appropriate service elements.

1.1.4 Supportability and Maintainability

Many developers need to reconsider their support and maintenance practices in light of web enablement. Traditional practices, such as distribution by diskette, allowed user bases to easily use different software versions and distribute interim releases and bug fixes on demand. NMCI and NEP sharply limit the number of different versions of the software that can be loaded into each environment and minimize the transition time. However, both save developers considerable effort in media distribution and tracking site updates because they can rapidly propagate software to all users and server hardware.

Even the flexibility inherent in web servers that allow developers to change applications on demand can be problematic. Changes that require modification to the user desktop (new plug-ins or mobile code) will require re-certification. In addition, substantial changes in the web site user interface may change the required interface code in the NEP and require re-certification. Developers must consider

the ability to make changes and the difficulty of making them when preparing their applications for migration to NMCI and NEP. Developers may release major revisions requiring certification once a year. Minor content changes on developer hardware or shared infrastructure may be made as needed.

1.1.5 Factors Affecting Web Enablement

For the Navy, "web enablement" means that the developer has followed the registration, development, integration/testing, and deployment processes in this document and the application has been approved for production use by the NEP IT Governance Board. This section lists factors that program managers and developers should consider when developing a web-enablement strategy.

1.1.5.1 Existing Web Applications

While having content on the WWW is not equivalent to web enablement, early adoption of web-based content and applications will accelerate migration to the NEP. The main issues that will influence the developer with regard to integration of a web application in the NEP are the following:

- Implementation of web technologies and associated standards as discussed in Section 3 (Design and Development).
- Presentation styling. The web application may conflict with NEP styling conventions. Portal look-and-feel integration may also be an issue, especially for applications using multiple frames.
- 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 NEP environment is left to the program, application, or content manager. It is strongly recommended that this entire document be reviewed prior to these undertakings.

1.1.5.2 Authoritative Data Sources

All authoritative non-tactical data sources are required to web enable. Developers are encouraged to use XML to return data through the provided web-enabled interfaces. XML will provide a universal baseline data transfer method for interoperability and data migration and allow developers to access authoritative sources across the enterprise.

Note that web enabling does not necessarily imply that these data sources would be visible to end users using a portal interface, although web enabling does allow a portal interface to be created. Authoritative data sources should be submitted and registered as Data Oriented Services (DOS) in the NEP Service Registry. This allows other developers to discover these services along with the necessary technical and contact information to make use of them. See 2.1.11.3 Service Registry Browser for more information on the browser.

While web enabling is a requirement for authoritative data sources, it is not expected that this will be the only access method. Alternate data transfer methods, including proprietary replication capabilities in many databases, may be required for high performance and close synchronization.

To maintain data integrity, many authoritative data sources are expected to be read-only upon initial designation and release. Developers and program managers who control authoritative sources are expected to maintain them in accordance with applicable laws and regulations in coordination with

FAM. Eventually, read and write access to sources should be provided with documented access controls and agreements delineating data responsibilities.

1.1.5.3 Real Time versus Non-Real Time

The web and Internet are not real-time meda. There is no intention of firing a weapon from a web browser. Real-time, rapid-response systems such as fire control systems are poor candidates for web enabling. However, status and historical information from real-time systems should be web enabled for integration in maintenance and command-and-control systems.

1.1.5.4 User vs. Administrator Roles

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 web enabling the end-user interface to deliver as much capability to the end user as resources and time permit. Rewriting existing management interfaces often has a cost higher than any benefit gained by the managers. New applications, however, should be web enabled entirely.

1.2 NMCI DESKTOP APPLICATION ANALYSIS

This section is intended to assist developers of applications that may fall under Table 2: Legacy Application Development Scenarios 3 through 6, which represent mostly desktop client/server models commonly known as "thick" client models intended to operate on NMCI. The "Preparation and Analysis, Phase One" section of Supplement A (the NRDDG) contains the latest background information for planning NMCI specific thick client software deployments. Several Navy Messages on legacy applications may apply.

1.2.1 Legacy Applications Transition Guide (LATG)

For the purposes of this guide, "legacy applications" refers to any customer software application that exists prior to the Assumption of Responsibility (AOR) and that is not included in the NMCI standard seat services or the Contract Line Item Number (CLIN) 0023 catalog. The NMCI Legacy Application Transition Guide (LATG) provides a detailed look at the processes used to transition DON legacy applications into the NMCI environment. The LATG is found at http://www.nmci-isf.com/legacy_applications_transition_guide.pdf . Please refer to the LATG for more information on current applicable legacy application analysis.

1.2.2 Navy Application Rationalization

Several emerging processes exist for Navy application rationalization, most of which are beyond the current scope of this document. Interested developers should follow instructions provided to them by their Echelon-level Functional Area Manager (FAM) and/or Functional Data Manager (FDM) point of contact (POC). Please refer to the DON Application and Database Management home page at https://www.dadms.navy.mil for the latest information on Navy application rationalization efforts... Please refer to APPENDIX D: Points of Contact for more information.

1.2.3 Information Strike Force Tools Registration

Developers of desktop applications must register with the current authoritative source for NMCI applications, the ISF Tools Database, available by following the "making the transition" link at http://www.nmci-isf.com or directly at https://usplswebh0ab.plano.webhost.eds.net/isftool/Login.jsp.

1.2.4 ISF Tools Database Description

The ISF Tools Database is the current authoritative database for NMCI applications. Application developers/owners must register to ensure their legacy, emerging, or new applications are rationalized, listed, and certified for NMCI to meet Navy enterprise standards and ISF requirements. The goal is to ensure applications are functionally necessary (rationalized) and appropriate for the NMCI environment. An application developer must request access to the ISF Tools Database and submit applications for NMCI certification.

Once ISF Tools Database access has been granted a CDA can do several things with the ISF Tools Database, such as:

- Check the status of certification
- View application survey data
- Add additional applications
- Follow a command rationalization process
- Submit applications
- View rationalized lists of applications and reports

These capabilities are based on the level of access granted by the Echelon POC. For more information, application developers should download and review the ISF Tools User Manual available in the help area of the ISF Tools Database or contact the ISF Tools Database POCs. Please see ISF POCs in APPENDIX D: Points of Contact for further assistance.

2. DESIGN AND DEVELOPMENT – PHASE II

Applications that have made it through the functional area rationalization of Preparation and Analysis – Phase I may proceed to Design and Development – Phase II. This section is broken down into two major sections specific to NEP integration and NMCI workstation/server environment integration.

2.1 NAVY ENTERPRISE PORTAL INTEGRATION AND TFWEB

The purpose of this section is to provide detailed information and guidance to developers interested in migrating content, applications, and services into the NEP at https://portal.tfw.navy.mil. To accomplish this objective, this section will provide an overview of the NEP implementation and provide specific examples to demonstrate the integration approaches that are discussed. The authors have sought to minimize the complexity of the integration process by viewing the NEP and the content, applications, and services to be integrated as "black boxes." Much of this guide focuses on providing detailed information about the interfaces between the "black boxes" vice describing in detail the inner workings of the NEP and its components.

2.1.1 NEP Scope

The scope of this section is limited and makes assumptions about the target audience and the level of knowledge within the developer community. References to web sites that provide detailed information about the technologies, standards, interfaces, and protocols used are provided in the list of references in APPENDIX B: References. Additional information, including frequently asked questions, detailed coding examples, and access to technical support, is available via the TFWeb Open Source Site at https://tfw-opensource.spawar.navy.mil/.

2.1.2 Assumptions

This guidance makes the following assumptions:

- Developers have already web-enabled their applications or obtained the knowledge necessary to web-enable their existing applications or decompose their applications into web services.
- Developers have already obtained the knowledge necessary for creating new web services.
- Providing this type of information is outside the scope of this guide. Including such information might inadvertently limit the flexibility and innovation that should be afforded to NEP developers.
- This guide has been updated to incorporate recent architectural changes to the NEP. These changes have not been fully tested and implemented. As a result, it is likely that updates to this document (in the form of errata sheets) will be published as required.

2.1.3 Target Audience

This guidance is intended for developers who will be directly involved in migrating content and application functionality into the NEP. It will most benefit those involved in the actual coding that will be required to complete the migration effort and the technical supervisors of the developers. It will have limited benefit to program managers.

2.1.4 Prerequisite Knowledge

In keeping with the scope of the document, this guidance assumes the target audience has some knowledge of the following areas:

- Basic understanding of the Department of Defense (DoD) public key infrastructure (PKI) initiative and DoD PKI certificates (see 2.1.10.1.3 Security Policy and PKI 3.0 Requirement).
- Working knowledge of design and development of web applications using industry standard tools and techniques (such as HTTP, HTML, and cascading style sheet [CSS]).
- Detailed knowledge of the content being migrated into the portal as well as the underlying design of the content.
- Working knowledge of XML and its related technologies (such as XML Schema and XSL).
- Working knowledge of web services-oriented architectures and their related technologies (such as Simple Open Access Protocol [SOAP], Universal Description Discovery and Integration [UDDI], and Web Services Definition Language [WSDL]).
- Basic knowledge of n-tier architectures, such as the common web 3-tier architecture (presentation, application, and data).

Before beginning detailed technical discussions, this guide presents background information that should be useful to developers who are unfamiliar with portal environments.

The W3C is the organization primarily responsible for developing and recommending technical specifications for the web community. W3C is an international organization that was founded in 1994 and is committed to ensuring that the web reaches its full potential.

XML developers shall only make use of W3C technical specifications holding the status of "Recommended." A "W3C Recommendation" is a technical report that is the result of extensive consensus-building inside and outside of W3C about a particular technology or policy. W3C considers that the ideas or technology specified by a Recommendation are appropriate for widespread deployment and promote W3C's mission (see http://www.w3c.org for further definition). XML-related standards promulgated by other nationally or internationally accredited standards bodies, such as: International Organization for Standardization (ISO), Institute for Electrical and Electronic Engineers (IEEE), American National Standards Institute (ANSI), Organization for the Advancement of Structured Information Standards (OASIS), United Nations/Centre for Trade Facilitation and Electronic Business (UN/CEFACT), Internet Engineering Task Force (IETF), should also be adhered to when developing applications within the domain that the standard addresses. When a standard produced by one of these bodies competes with a similar product of the W3C, the W3C standard shall take precedence.

A web developer should be familiar with the following standards:

- HTML
- CSS
- XML
- XSL
- XHTML
- ECMAScript (JavaScript)
- DOM

Complete information about these standards, including specific versions that are required for the NEP, is listed in Table 24: Important Standards.

2.1.5 Getting Started with the Navy Enterprise Portal

Much of this guide is focused on the technical requirements that developers need to understand in order to integrate their applications and content into the NEP. The following background information is provided as a way of obtaining a common understanding of concepts, terms, and the NEP architecture prior to discussing the more detailed aspects of integration requirements.

A Change Control Board (CCB) controls changes to the NEP. Relatively small changes and bug fixes are reviewed and approved on a weekly basis. Once approved, these changes are developed and deployed within a few days or, in rare cases, a few weeks. The NEP architecture document will be updated approximately twice a year to incorporate these changes and any enhancements added to the architecture. While many of these updates will be considered minor, it is possible that major changes to the architecture will occur on an infrequent basis, thus coordination with TFWeb and version control is critical to ensure consistent and reliable web services.

The current NEP architecture is Version 3.0. NEP v3.0 is has been implemented at various NEP afloat and ashore sites. This document describes development for the NEP architecture, NEP v3.0. This architecture is discussed in detail in the "Web Enabled Navy Technical Architecture Description" document, and is downloadable from the TFWeb Open Source Site https://tfw-opensource.spawar.navy.mil/.

2.1.5.1 Concepts

2.1.5.1.1 Portals

A portal is a web server that provides a secure, single point of interaction with diverse information, business processes, and people and is personalized to a user's needs and responsibilities. Portals generally provide the following capabilities:

- A single point of access to all resources associated with the portal domain
- Personalized interaction with the portal services
- Federated access to hundreds of data types and repositories (aggregated and categorized)
- Collaboration technologies that bring people together
- Integration with applications and workflow systems
- Multiple workplaces

Figure 3: Tabbed Workplaces, User Profile Access, and InfoStore Components and Figure 4: InfoStore Contents (Channels and Library) highlight several of the common components of portals.

The "Workplace" is a personalized view of the content portal users have accessed. Content and services are placed onto the Workplace by dragging library or channel content objects onto the Workplace area. Users have the ability to create multiple Workplaces. These Workplaces can be created to support virtual interest groups and communities of interest.

The "InfoStore" is a repository that contains information that the enterprise wants to make available to portal users. It is composed of a Library of content objects like hyperlinks, documents, links to other applications, and Channels, which are logical groupings of this information.

A user "Profile" controls the way the NEP looks and how it organizes the user's information.

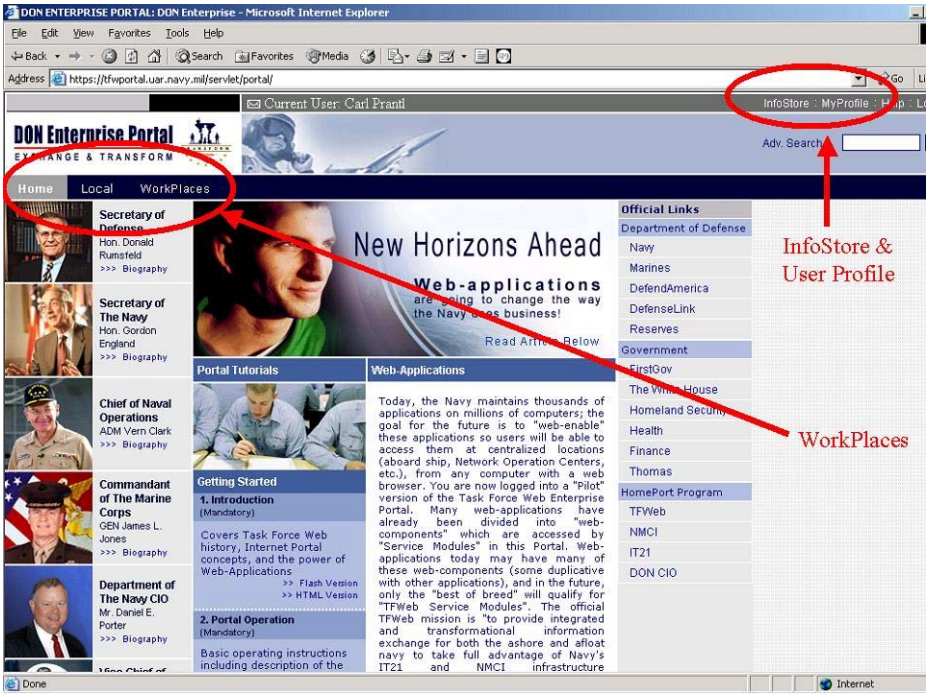


Figure 3: Tabbed Workplaces, User Profile Access, and InfoStore Components

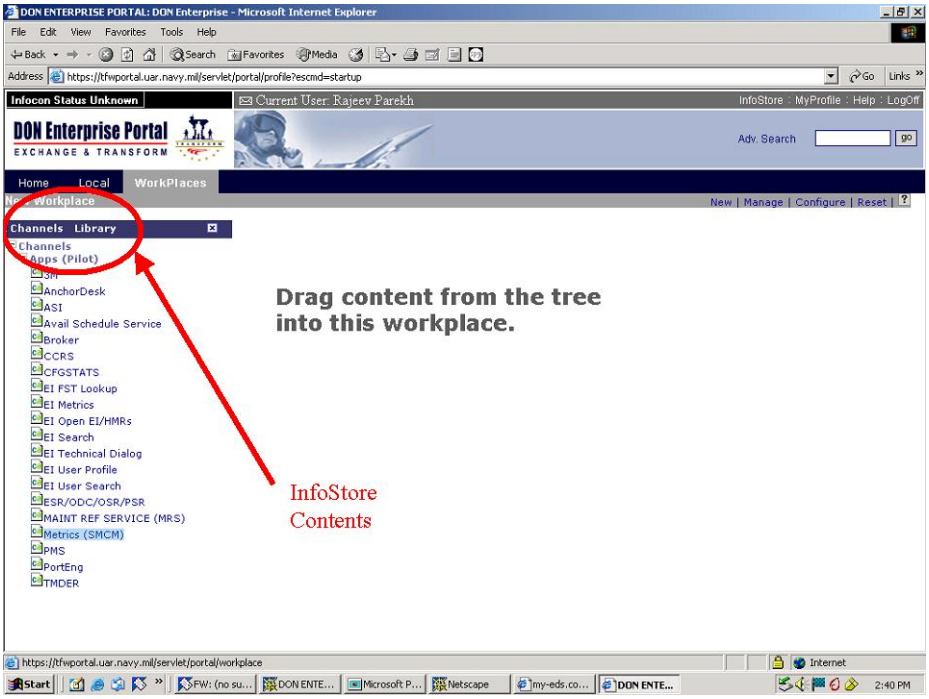


Figure 4: InfoStore Contents (Channels and Library)

2.1.5.1.2 Portlets

A portlet is a visible, active window that end users see within the NEP. Portlets contain the rendered output generated by developers and content providers. Figure 5: Portlet Examples illustrates several instances of portlets.

Several portlet types can be created to support portal integration. These types are listed in the following text and are fully discussed in 2.1.6 Portal Content Integration.

Reference Portlet
External Content Integration Portlet
Content Integration Portlet

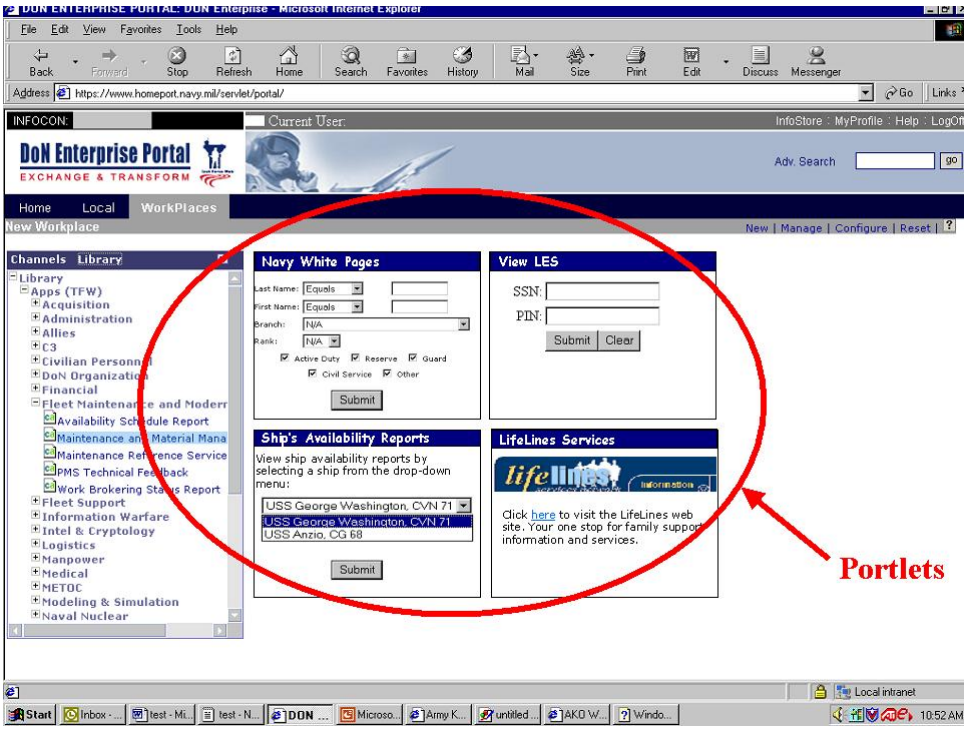


Figure 5: Portlet Examples

2.1.5.1.3 Services

Services are software components that are in the NEP's service registry and provide access to content, execute business logic, or expose application functionality. There are two types of services that are created for integration into the NEP. These services are listed in the following text and defined in Appendix A: Glossary.

- User Facing Services (UFSs)
  A software component that receives a UFS Request from the portal and returns a UFS Response that formats the content for display (usually in a markup language such as HTML or WML) to produce visual output in a portlet.

- Data Oriented Services (DOSs)
  A software component that receives a request and optionally returns an XML Data Response to a UFS or another DOS. A DOS has no visual or presentation component.

This document does not provide detailed information on the development of DOSs. These details will be included in later versions of the document as standards mature. Authoritative data sources should be submitted as DOSs and they will be registered in the NEP Service Registry. This allows other developers to discover these services along with the necessary technical and contact information to make use of them. Also, see 2.1.11.3: Service Registry Browser.

In general, an "application" consists of one or more "services". Each service is exposed through a well defined interface providing isolation of implementation details. To allow a NEP user to connect, at least one of these services must be a UFS.

2.1.5.1.4 Three-Tiered Architectures

A three-tiered architecture is a layered architecture that supports the development of robust, scalable applications that present information in a web-based environment. The architecture seeks to separate the application into three distinct tiers: presentation, business logic, and data. Figure 6: Three-Tier Architecture and Portal illustrates the relationship of the portal and portal services to the layers of the architecture.

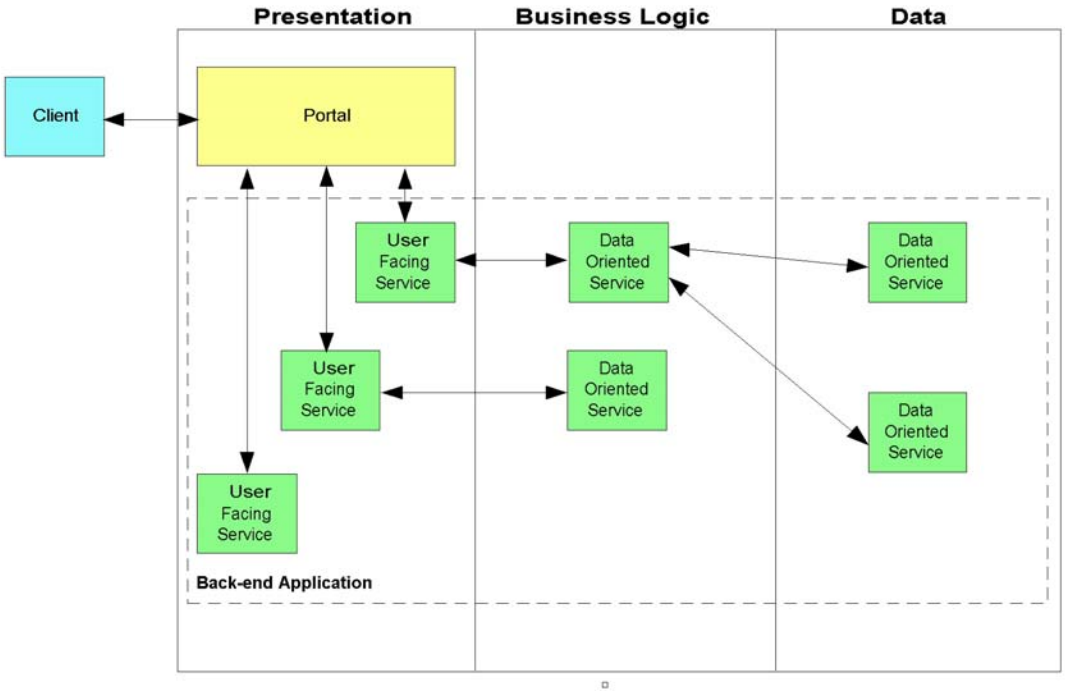


Figure 6: Three-Tier Architecture and Portal

2.1.5.2 Navy Enterprise Portal

2.1.5.2.1 Description

The NEP provides a common web infrastructure across the enterprise, both ashore and afloat.

From the user's perspective, the primary component is the portal. Users will use a portal client (browser, personal digital assistant [PDA], etc.) to connect to the portal using a common hyperlink across the enterprise, https://portal.tfw.navy.mil. Developers should understand that there are multiple physical instances of the portal throughout the Navy. The current architecture envisions portal instances aboard each ship, the NMCI NOCs, the BLII NOCs, and each of the Fleet NOCs. The NEP will capture all access requests and direct them to the nearest available physical instance of the portal.

UFSs are integrated with the portal to provide portlet content in the portal. The UFS consists of a variety of web applications and web services. The developer will provide access to one (or more than one) UFS as applicable to the application. The UFS will reside outside the NEP and can be hosted in a number of places to include an afloat Fleet application server and an NMCI web server. However, typically the UFS would be co-hosted with the backend web application or service.

In some cases, one UFS may be hosted in many physical locations. An example is a shipboard web application with a UFS that exists on many ships. The UFS is certified and registered once for use within the NEP, but with many physical instances, the application owner must provide separate deployment metadata for each instance. Typically, this is a separate URL and point of contact for each instance with all other metadata in common. Each instance of each service is listed in the registry as a separate service. In this case, the service description should also include the physical location of the service.

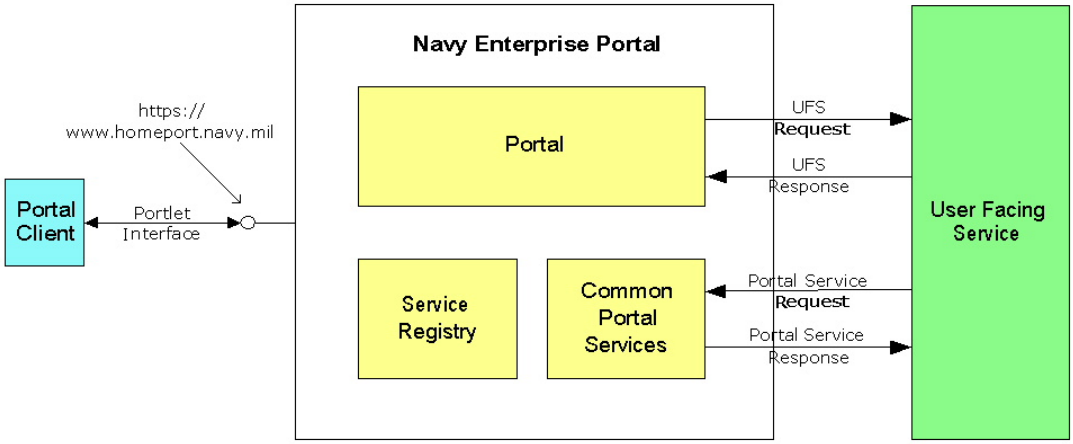


Figure 7: Navy Enterprise Portal

2.1.5.2.2 Components

The components of the NEP are portal, service registry, and the common portal services. This section describes each component.

2.1.5.2.2.1 Portal

The portal engine provides user access to the content from the distributed UFS. The portal aggregates this content in portlets and presents it to users.

2.1.5.2.2.2 Service Registry

The service registry is a private, globally distributed registry of web-application and web-services information. All web applications and web services are required to provide metadata for the registry. The registry contains UFS and DOS metadata for all services that are accessible by the NEP and potentially by other applications. Each physical instance of service is entered separately into the registry. There exists one service registry per NEP instance. Each of these replicates service metadata with other service registries across the enterprise to provide a common enterprise service registry. The service registry has been implemented as a UDDI 2.0 registry.

2.1.5.2.2.3 Common Portal Services

The portal provides common services that may be used as needed by the UFS. The use of the common portal services is optional. Refer to 2.1.9 Portal Services Interfaces for more information about these services.

2.1.5.2.2.4 Portal Clients

The NEP is capable of supporting many different types of clients. While the current implementation of the NEP renders all XML/XSL into HTML for delivery, future portals will deliver content according to the type of device used to load the page, the role of the user connecting to that page, and the data type being displayed. Dynamic rendering will be achieved by applying XSL style sheets relevant to the device used. Figure 8: Support for Multiple Clients and Dynamic Content Rendering illustrates this concept.

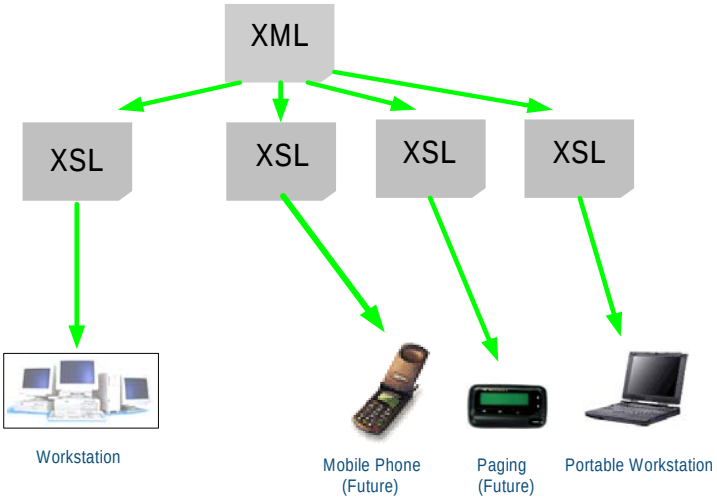


Figure 8: Support for Multiple Clients and Dynamic Content Rendering

XSL style sheets will be used to define the access for each communication channel. Data will be formatted in XML by portal compliant applications or transformed into XML format by the integration

framework prior to portal access. The style sheets allow this common data to be formatted based upon user context.

The presentation layer of the application should be designed with display and user input device independence in mind and is required to support the presentation standards supported by the NEP (e.g., appropriate D/HTML, XML, and WML versions).

2.1.5.2.3 Interfaces

2.1.5.2.3.1 Portlet Interface

This interface is used by portal clients when accessing the aggregated portlet UFS components through the portal. Please see 2.1.7 Portlet Interface for more information.

2.1.5.2.3.2 User Facing Services Interface

This interface is used by the portal to communicate with UFS. Refer to

2.1.8 User Facing Service Interface for more information about this interface.

2.1.5.2.3.3 Portal Services Interfaces

These interfaces are used by the UFS when accessing the portal services. Refer to 2.1.9 Portal Services Interfaces for more information about these services and their interfaces.

2.1.5.2.4 NEP Execution Sequence

Figure 8 illustrates the processing sequence of the NEP implementation. Across the top are the component interactions. Moving down the diagram is the order of interactions.

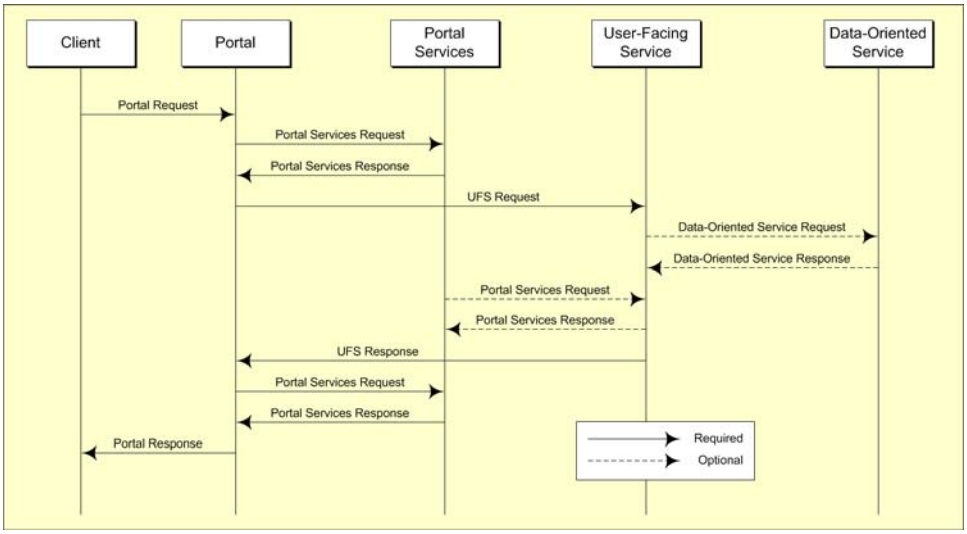


Figure 9: NEP Execution Sequence Diagram

2.1.5.3 Standards

Initial lists of standards, including specific versions that are required for the NEP, are listed in Table 24: Important Standards (all external links). Additional guidance about XML standards is under development by the DON XML Work Group (DONXML WG) and can be found at http://quickplace.hq.navy.mil/navyxml. The DONXML WG seeks to provide a single authoritative source for issues related to implementation and fielding of XML throughout the Department

2.1.5.3.1 Why Web Standards?

With its tremendous growth, the web needs standards to realize its full potential. Web standards ensure that everyone has access to the same information. Future use of the web will not be possible without worldwide standards.

The advantages of standards-based development are numerous:

- Development and maintenance time are shortened because several versions of code are not required to accomplish the same result.
- Faster support is provided for new hardware (like mobile telephones and other handheld devices) and new software (like micro-browsers).
- Web development teamwork is simplified, because it is easier for the developers to understand each other's coding.
- Standardization can increase access to sites. Audiences are not limited to specific browsers.
- Standard web documents are easier for search engines to access, easier to index more accurately and easier to convert to other formats.

2.1.5.3.2 Proprietary Extensions

XML implementations shall not make use of proprietary extensions to XML-based specifications.

2.1.5.4 Integration Planning Considerations

During the process of integrating content with the NEP, several decisions by the developer and content provider are required. The first, and most important, decision is the type of integration to be provided. The three integration types are:

- Reference
- External Content
- Content

Table 3: Required Implementation Items by Integration Type provides a detailed description of each type of portal integration.

Other decisions involve the use of optional, portal-supported features. These features may be needed to support unique requirements of the content provider's application or to meet certain application-specific goals.

2.1.5.4.1 Selection of Integration Type

In general, the Navy expects content integration, although valid reasons may exist for choosing either external content or reference integration. Please note that some cases using external content or reference integration, as opposed to content integration, may require justification.

To fully understand the impact of selecting the different types of content integration, the later sections of this document must be read and understood; however, to assist developers in planning and designing their integration, a summary of the required implementation items for each integration type was created. These items are shown in Table 3: Required Implementation Items by Integration Type. Please note that only required implementation items are shown in the table. Optional items are described in the next section.

2.1.5.4.2 Optional Portal-Supported Features

The following optional features may be called by developers to extend portal functionality. Availability of the features is controlled by metadata provided by the developer at the time a UFS is submitted. Please refer to Portlet Interface for a description of this metadata.

The optional features include the following:

- Sending the portal context metadata to the UFS (See APPENDIX E: PRI Data).
- Rewriting URLs in UFS output (See APPENDIX H: URL Rewrite Guidelines).
- Identifying whether the UFS is implemented using a SOAP or HTTP interface.
- Returning errors to the portal from the UFS (See 2.1.8.3.4: UFS Response)
- Choosing to use a standard portal cascading style sheet (CSS), even when not required (See APPENDIX F: Portal CSS).
- Using the Common Identity in the UFS.
- Re-authenticating the portal user in the UFS (Special considerations apply. See APPENDIX G: Application Security.).
- Maintaining the state of a portlet across calls to the UFS (See Section 2: Portlet Interface).

Table 3: Required Implementation Items by Integration Type

Integration Type	Required Implementation Items
Reference	The UFS must be available on a host that allows network access by the portal. The URL of the UFS must be provided to the Navy Enterprise Portal team, and the URL must be resolvable to an IP address using a domain name service (e.g. DNS) available to each of the portal instances that will access the UFS. Contact information for UFS must be provided to the NEP team.
External Content	All items listed for Reference integration. All UFS output must be delivered to the portal (instead of directly to the client). UFS output must include properly rewritten URLs before being delivered to the client. (See APPENDIX H:URL Rewrite Guidelines for technical details.) The UFS may create the correct URLs or, optionally, the portal URL rewrite feature may be used. UFS output must be "portal friendly." (See Section 2.1.6 Portal Content Integration for details.)
Content	All items above for external content integration. A standard portal cascading style sheet (CSS) must be used to format all UFS output. (See APPENDIX F: Portal CSS.) Resizable portlet panes for displaying content must be fully supported.

2.1.6 Portal Content Integration

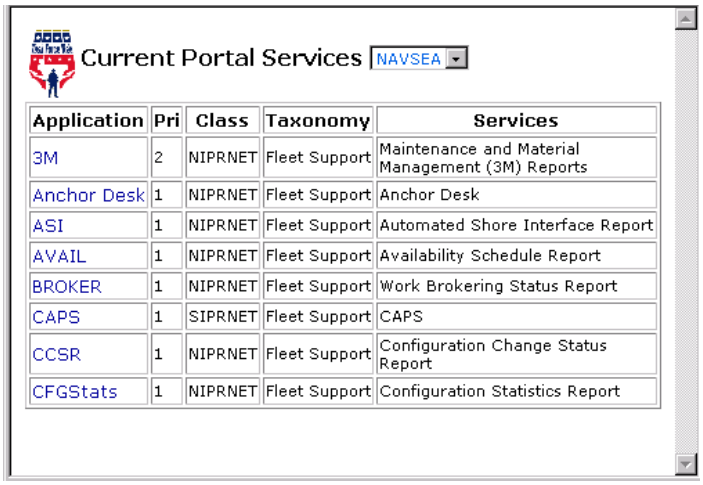
Integration of content into the NEP involves generation of a portlet interface to the application function, data, or documents to be made available to portal users. Technically, this is accomplished through creation of a UFS. The output of the UFS will exhibit the characteristics of portlets described in this section.

The UFS is responsible for managing the interaction between the portal and the back-end application. The UFS is called by the portal. The contract governing this interface is described in User Facing Services Interface. All interaction with the portal is stateless. Management of state between invocations of the UFS in a conversational interaction with a user is the responsibility of the UFS and back-end application.

There are various types of interaction possible within the scope of a portlet. First, the simplest form is the presentation of static content within a portlet frame. The portal will call the UFS and display content returned in the portal frame. Second, increasing in complexity, a form can be displayed that will post to a backend application. The resulting response can be displayed within the same portlet frame or directed to a new browser window.

Portlets that need automatic periodic updates, such as for updating news feeds, may use client side script (such as JavaScript) to refresh the content for a single portlet. Each portlet frame is assigned a separate session ID, which may be passed from the portal to the UFS in the Portal Request Interface (PRI) request block, as described in 2.1.8.3.2 UFS Request.

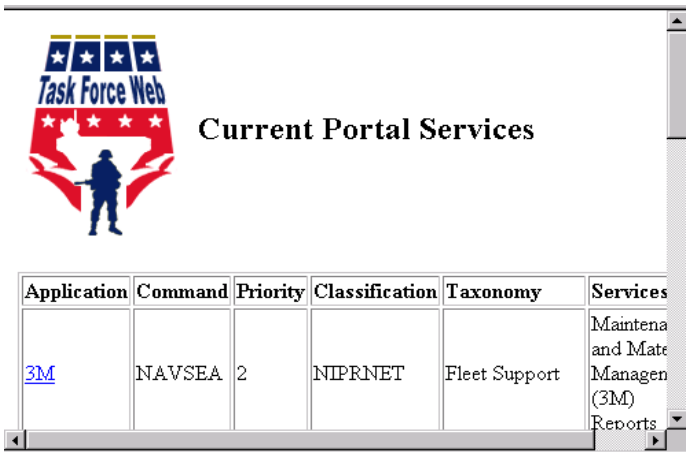
Portlet development is guided by the ability of an application to provide content through the UFS that is compatible with user expectations for interaction with the content. This concept is often described as "portal friendliness." A user will rarely expect a portlet to interfere with the operation of the portal or other portlets on the portal workspace. Unexpected side effects or intrusive user interaction should be avoided. In addition, a well-designed portlet will adapt to the user's profile customizations and window-sizing preferences. See Figure 10: Good and Bad Portlet Examples for examples of a "good" portal-friendly portlet and a "bad" poorly designed portlet. Essentially, a "good" integration presents content in a pleasing visual ergonomic and functional way while a "bad" portlet may not display correctly upon resize, may have a non-standard look and feel, or require excessive scrolling.



Current Portal Services NAVSEA

Application	Pri	Class	Taxonomy	Services
3M	2	NIPRNET	Fleet Support	Maintenance and Material Management (3M) Reports
Anchor Desk	1	NIPRNET	Fleet Support	Anchor Desk
ASI	1	NIPRNET	Fleet Support	Automated Shore Interface Report
AVAIL	1	NIPRNET	Fleet Support	Availability Schedule Report
BROKER	1	NIPRNET	Fleet Support	Work Brokering Status Report
CAPS	1	SIPRNET	Fleet Support	CAPS
CCSR	1	NIPRNET	Fleet Support	Configuration Change Status Report
CFGStats	1	NIPRNET	Fleet Support	Configuration Statistics Report

Good



Current Portal Services

Application	Command	Priority	Classification	Taxonomy	Services
3M	NAVSEA	2	NIPRNET	Fleet Support	Maintena and Mate Managen (3M) Reports

Bad

Figure 10: Good and Bad Portlet Examples

Depending on the capabilities of the application that provides the content through the UFS, a portlet developer must select the type of integration that best supports user interaction with the content in a user-friendly manner. The following general categories of portlet integration have been defined to illustrate the different characteristics of good portlet design.

2.1.6.1 Reference Integration

A portlet using reference integration is simply a portlet that contains information about the content available through the UFS, a hyperlink to access the content, and optionally, lightweight graphics. The user is presented with a new window in which the application content is displayed. This supports full-screen interaction with the content separately from the parent portal window. This type of portlet integration is generally used when the content is not portal friendly or the nature of the application or service is not usable within a portlet frame. Figure 11: Example of a Portlet Using Reference Integration is an example of a portlet using reference integration.

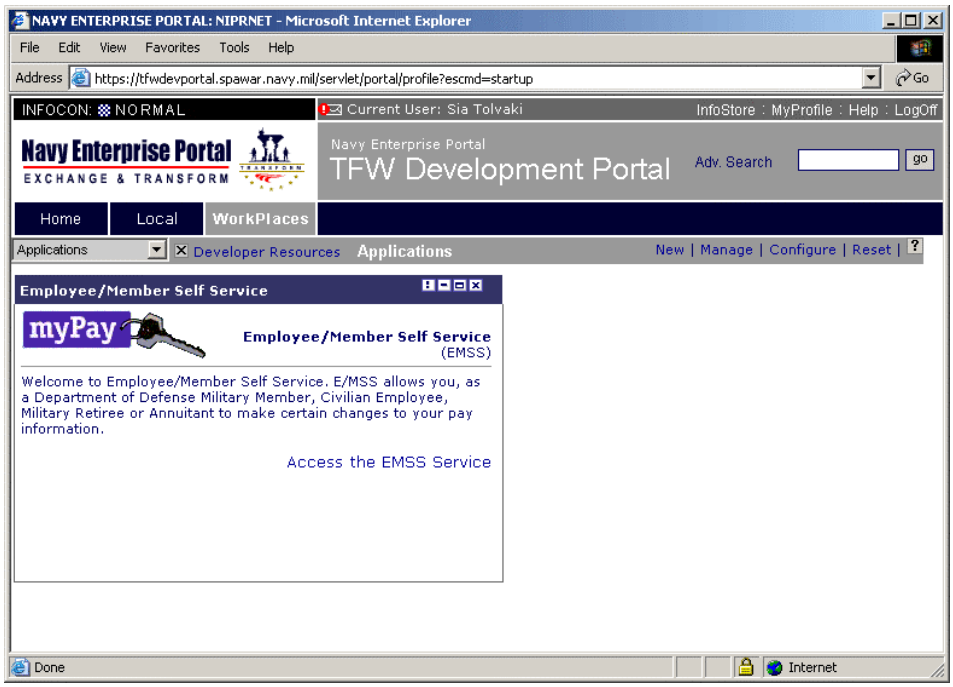


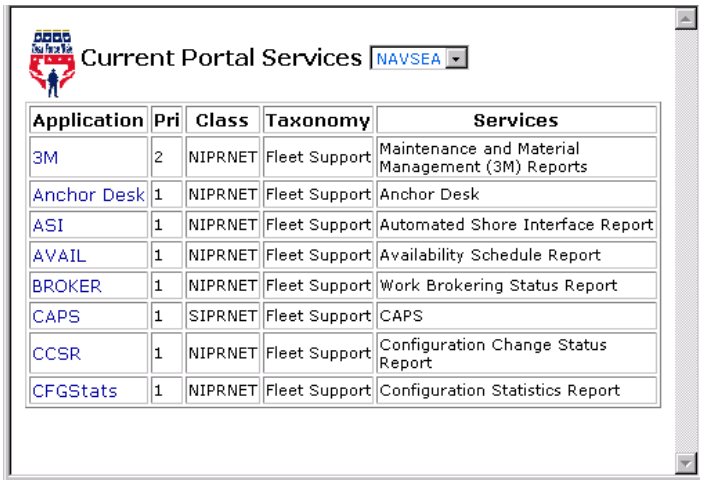
Figure 11: Example of a Portlet Using Reference Integration

A portlet using reference integration provides a portal user a means to access the content of the existing application. When a user clicks on the "Access the EMSS Service" hyperlink, the existing web site or web application is presented in a new browser window. It then becomes the responsibility of the application owner to ensure that the web application is physically accessible to enterprise users. If the content is hosted within an enclave that restricts resolution of the URL from the client, the portal proxy URL rewrite capability will not be available to the content window opened from the portlet hyperlink. A user accessing the content from outside the hosting enclave will receive a "Cannot find server or DNS Error" message.

The backend application service or content owner hosts the UFS. In this example above, the UFS is a simple HTML file. For a more detailed look at this reference integrated portlet see APPENDIX J: Case Study 1: Employee/Member Self Service Integration.

2.1.6.2 External Content Integration

A portlet using external content integration is a portal-friendly connection to application content or services. It is differentiated from a portlet using reference integration by this portal friendliness. Data content or service interaction is provided in the portlet, allowing user interaction with the content through the portal frame. A key differentiator of a portlet using external content integration is that the look and feel and content rendering are controlled by the backend application content provider, as opposed to a portlet using content integration (see 2.1.6.3) that delegates look-and-feel control to the portal engine. A portlet using external content integration is shown in Figure 12: Example of a Portlet Using External Content Integration.



Current Portal Services NAVSEA

Application	Pri	Class	Taxonomy	Services
3M	2	NIPRNET	Fleet Support	Maintenance and Material Management (3M) Reports
Anchor Desk	1	NIPRNET	Fleet Support	Anchor Desk
ASI	1	NIPRNET	Fleet Support	Automated Shore Interface Report
AVAIL	1	NIPRNET	Fleet Support	Availability Schedule Report
BROKER	1	NIPRNET	Fleet Support	Work Brokering Status Report
CAPS	1	SIPRNET	Fleet Support	CAPS
CCSR	1	NIPRNET	Fleet Support	Configuration Change Status Report
CFGStats	1	NIPRNET	Fleet Support	Configuration Statistics Report

Figure 12: Example of a Portlet Using External Content Integration

2.1.6.3 Content Integration

A portlet using content integration is similar to a portlet using external content integration in that the content is designed and intended to be displayed through the shared workplace of a portal. The differentiating characteristic of full content integration is the delegation of look and feel to the portal. This is accomplished through incorporation of the portal look-and-feel CSS stylesheet guidelines found in APPENDIX F: Portal CSS. The CSS style sheet reference that is associated with the template a user has selected can be extracted from the PRI data request as described in APPENDIX E: PRI Data.

The benefit of incorporating the portal user's specified look and feel is best illustrated by the example of an application that may be used in the darkened conditions of a ship's bridge (See Figure 13: Example of a Style Sheet Changing). A user needs to protect his or her night vision by selecting a template with dark backgrounds and low-contrast text. Contrast is defined as the degree of color differentiation between an object on the screen and its background color. If a portlet is integrated, the UFS and the portlet content are re-rendered using the new CSS-defined styles. Another possible benefit is the incorporation of high contrast styles for compliance with Section 508 (see http://www.section508.gov http://www.section508.gov/). In addition to integration of the look-and-feel styles, a portlet using content integration will fully support content display within a resizable portlet pane. Table display is one area where assumption of pane size can greatly affect the readability of the content. Also, content that is substantially larger than the space available in a portlet frame forces the user to scroll to get to all of the content and makes the portlet less attractive for use.

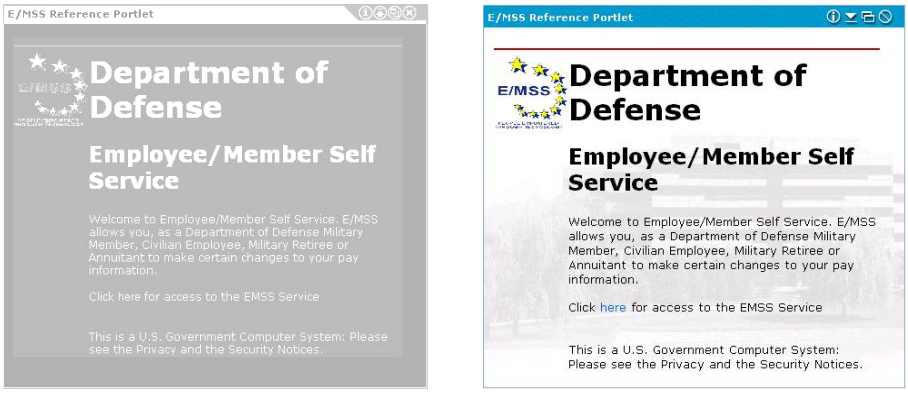


Figure 13: Example of a Style Sheet Changing

2.1.6.4 Portlet Characteristics

Table 4: Portlet Content Integration Characteristics is intended to encapsulate the characteristics that differentiate the types of portlet integration described here. It is not intended as a compliance rating scale. Unless otherwise indicated, the characteristics described in Table 4: Portlet Content Integration Characteristics are not required, but are elements of a complete portal integrated solution. Various factors will influence the ability of an application to implement each of the recommended characteristics, and it is the responsibility of the backend application developer to determine the most appropriate solution.

Table 4: Portlet Content Integration Characteristics

Portlet Characteristic	Portlet Integration Types		
	Reference Integration	External Content Integration	Content Integration
UFS Output	HTML or XML/XSL		
Portal Look-and-Feel Integration	Recommended	Recommended (consists of PRI "CLIENTSTYLE" style-sheet reference and portal-specific CSS tags). See style-sheet reference.	Required (consists of PRI "CLIENTSTYLE" style-sheet reference and portal-specific CSS tags). See style-sheet reference.
Compatible with Portal Reverse Proxy (URL Rewrite)	Recommended	Required, see APPENDIX H: URL Rewrite Guidelines.	
HTML BASE TAG for relative references vs. absolute references	Recommended	As required, see APPENDIX H: URL Rewrite Guidelines.	
Portal Rendering of XML/XSL to HTML (XSLT)	Available as an external service call only (see section 2.1.9.2.4).	Available using the in-line XML Transformation (see section 2.1.7.4.3) or using the XML Transformation Request (see section 2.1.9.2.4) as an external service call.	
Portal-Controlled iFrame compatibility	Recommended	Required.	
Mobile Code (Applets, ActiveX)	Allowed within Navy policies and guidelines. See APPENDIX N: Navy Mobile Code .		
Client-Side Script	Supported	Supported with the following restrictions: Frame refs cannot refer to '_top' frame. No dynamically generated URL links that aren't relative to the HTML BASE tag.	
Application-Controlled Frames / iFrames	Supported	Supported with the following restrictions: Cannot target 'top' frame. Frame refs should be named.	
Popup child windows	Supported	Supported in response to user interaction within the frame that launches the child window.	
UNICODE Support	Recommended		

2.1.7 Portlet Interface

The purpose of this section is to provide a concise, understandable description of the interface between the portal client and portal. (See Figure 14: Portlet Interface Section Scope.) An application developer defines the criteria that the portal uses to call the UFS when content is integrated into the NEP. This section provides information describing the context in which the interface is used, as well as the interface message and behavior.

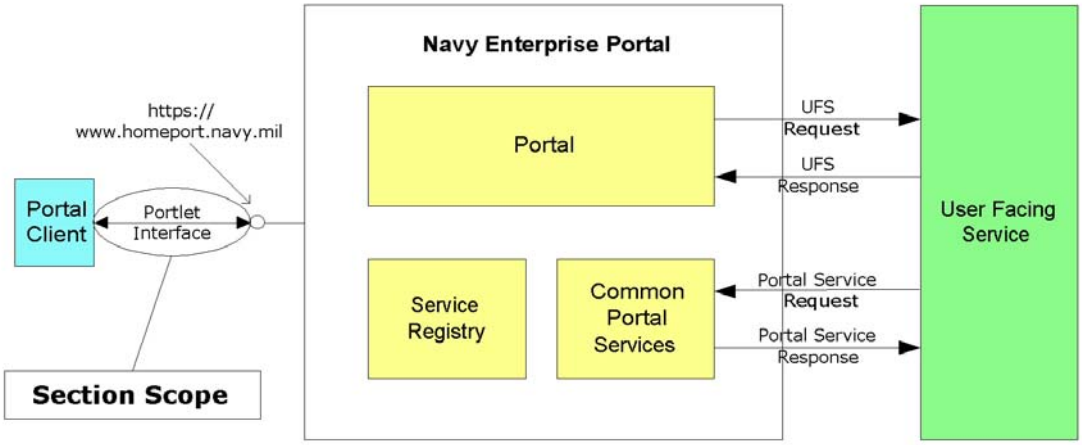


Figure 14: Portlet Interface Section Scope

The portal actually has two interfaces to the client. The first is the normal portal interaction with the client that manages personalization, aggregation, and customization. The interface described here is the interface used to call each of the portlet UFS instances aggregated on a workspace. The interface descriptions provided in the following text include the description of all valid options; however, the appropriateness of one option vs. another option is not discussed.

2.1.7.1 Interface Usage Context

The portlet interface context must be described, to fully understand of its usage. The context is best understood by describing how the portal uses metadata. UFS metadata is especially important, because this metadata is used as the input criteria (arguments) to this interface.

2.1.7.1.1 UFS Metadata Stored by the Portal

To enable content integration, the NEP stores several metadata elements for the UFS. These metadata elements are stored in the service registry. When a UFS is entered into the service registry, a unique identifier is assigned to each UFS. This unique key, called the serviceKey, is also included as a parameter on the URL.

With the exception of the serviceKey, UFS developers may supply the values of these metadata elements as they control how the portal calls the service. If an element is not supplied, then default

values are assigned at the time the service is entered into the registry. Please see Table 5: Portlet Service Request for more information about this metadata.

2.1.7.1.2 How UFS Metadata Is Used

The portal generates HTML pages to send to the client. Some of these pages contain a special hyperlink that refers to the UFS. Selecting (clicking) this hyperlink causes the UFS to be invoked. The UFS is not invoked directly. The portal will call the UFS on behalf of the client as described in the following text.

The portal-generated hyperlinks that refer to the UFS include the serviceKey of the UFS. The following is an example of a URL that may appear in a portal page:

https://portal.tfw.navy.mil/nep/PortalConnector?serviceKey=0A837A83-D8DE-43F4-A281-EC2981CB03A8 (sample link, not active)

When a hyperlink such as the above is selected (clicked), the portal is invoked on behalf of the UFS. The portal calls the UFS after performing any necessary tasks (as described later in this section). The URL used for calling the UFS does not include the serviceKey parameter described above. The following is an example of a URL that may be used by the portal to call a UFS:

https://www.application.navy.mil/user-facing-service/index.jsp (sample link, not active)

Note that the HTTP message sent to the UFS may also include PRI data (seeAPPENDIX E: PRI Data) and the user's Common Identity. The PRI data also includes some of the parameters that were retrieved from the service registry for the UFS. Note that this data is only available if the UFS developer submits service metadata that requests PRI data be sent to the UFS.

2.1.7.2 Interface Documentation Approach

The approach used to describe the portlet interface is frequently used, but may not be familiar to all readers. Therefore, a short explanation is provided in this section.

The description is composed of five parts:

- Preconditions define all of the requirements that must be satisfied before a request can be sent to the portal. These preconditions may include both real-time software execution conditions and policy conditions.

- Portlet Service Request describes the different types of requests that can be sent to the portal interface. Service developers must create service responses that meet the conditions in the section. Content and formatting of the messages are also described, although lengthy descriptions are usually referenced instead of being included in the following text.

- Portlet Service Responsibilities describe the tasks that need to be performed by the portal. These tasks may differ depending on the type of message received and the values of fields within the message.

- Portlet Service Response describes the different types of responses that are produced by the portal interface. Both content and format are described in this section.

- Post conditions define all the requirements that can be assumed to be met after a portal call.

The application developer should be familiar with all five parts of the interface description in order to properly utilize the common service. Detailed information for each part is provided in the following sections. (See APPENDIX E: PRI Data.)

2.1.7.3 Portlet Interface Overview

The portlet interface is the common service that is used to provide the interface between the portal and the client to call a UFS. All services called by the portal are referenced through this interface. The component is a servlet that provides consistency in the way UFSs are called, no matter what product is used to implement the portal functionality. The interface to a UFS is implemented based on industry standards where possible and provides additional functionality required by the NEP that does not yet have a standards-based implementation. As standards evolve, the portal will allow the Navy to control the migration to standards-based support. As portal standards mature and the interfaces are standardized, the portlet interface should be able to be deprecated.

The function of the portlet interface is twofold. First, it provides an abstraction layer to isolate the portal from the application interfaces. Second, it enables the set of transformation services necessary to facilitate communication and data exchange between the portal and a UFS.

2.1.7.4 Portlet Interface Definition

2.1.7.4.1 Preconditions

The portlet interface is invoked using the parameters and metadata supplied to the portal administrator as part of the service registration package. While the backend application does not actually call the portal, the parameters are based on the metadata supplied and greatly influence the manner in which the UFS will be called. Invocation from any source other than the portal is not supported.

The processing prior to invocation of the UFS by the portal is outlined in Figure 15: Portal Precondition Processing.

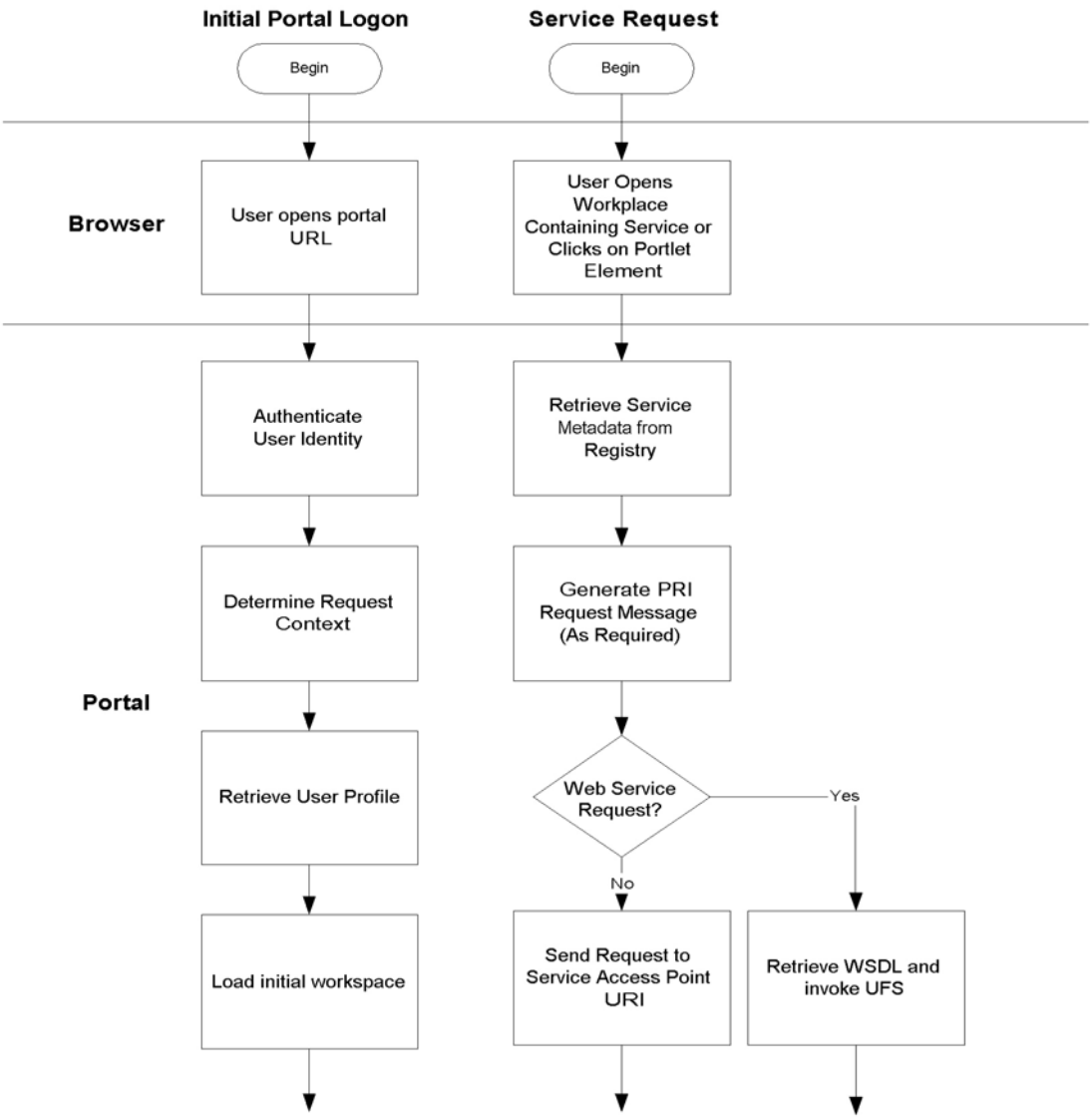


Figure 15: Portal Precondition Processing

Transmission Control Protocol/Internet Protocol (TCP/IP)-based connectivity is required from the portal to the UFS host.

2.1.7.4.2 Portlet Service Request

The portlet interface is callable from the following URL from all NEP instances:

https://portal.tfw.navy.mil/nep/PortalConnector?<parameters> (sample link, not active)

The available parameters for this interface are shown in Table 5: Portlet Service Request. Any <parameters> that are specified on the HTTP query string request will override the corresponding parameters for the service in the service registry.

Table 5: Portlet Service Request

Parameter	Values	Description	Required
ContentMimeType	[mime-type]	Helps determine how the portal should direct the processing of the UFS Response content. Providing the contentMIMEtype as metadata when registering the service will also allow for faster processing by the portal, but will not override the value in the Content-Type HTTP Header of the UFS Response.	N
GenerateCookie	[Y | N]	Controls whether the portal will generate a session cookie with the information necessary to allow re-invocation. This is intended for use when a forward proxy is used instead of the URL rewrite proxy.	N
insert Style	[Y | N]	Inserts the appropriate portal CSS reference into the HTML UFS output before sending to the portal client.	N
operation	[operation-name]	Used for WSDL invocations. The operation name in the WSDL file to be executed.	N
operationMessageParts	[MessagePartName1 ~MessagePartValue 1#MessagePartNam e2<MessagePartVal ue2] Note: The "~" character is used to delimit MessagePartNames and MessagePartValues. The "#" character is used to delimit the pairs of MessagePartNames and MessagePartValues	Used for WSDL invocations. The messages pass to the operation to be executed. The message parts for the input message to the service's operation are as defined in the WSDL file. The keyword MessagePartName will be validated against the message part names in the WSDL file. The MessagePartValue will be passed as the parameter value to the service's operation. For any required input message parts for which a value is not specified, a page will be generated requesting the information from the user. See the WSEE service for more information.	N
renderXML	[Y | N]	Controls whether the portal will attempt to render XML using an XSLT style sheet reference embedded in the XML document. Setting this to N will allow a service to pass the raw XML to the client to support client-side rendering.	N
rewriteURL	[Y | N]	Controls whether the portal will attempt to rewrite URL references in the return stream to proxy all requests back through the portal. See APPENDIX H:URL Rewrite Guidelines for more information about this feature.	N

Parameter	Values	Description	Required
sendIdentity	[Y | N]	If the sendIdentity argument is set to Y and the user is successfully authenticated, the portal will pass the user's Common Identity in a standard HTTP Header of the HTTP request that will be generated by the portal and sent to the UFS. The HTTP header parameter is named "NEPCommonIdentity".	N
sendPRI	[Y | N]	Indicates whether or not the PRIdata message is added to the request sent to the service. The method used to pass the PRIdata is dependent upon the type of invocation used. The PRIdata message is attached to the HTTP header for standard HTTP calls and, if SOAP is used, is embedded in a SOAP header. See APPENDIX E: PRI Data for more information.	N
serviceKey	<GUID>	The SERVICEKEY is the bindingKey from the NEP service registry that identifies the service to be invoked.	Y
UddiURL	[URI]	Used for WSDL bound service invocations. The default behavior of the WSEE service invoked by the portal is to look up all bindingKey references in the NEP enterprise registry. If the service is defined in a public UDDI registry, the Uniform Resource Identifier (URI) to the Inquiry API interface must be specified here.	N

2.1.7.4.3

2.1.7.4.4 Portlet Service Responsibilities

The portal will perform the following activities:

- The portal is called with the service key and any service-specific parameters.
- The metadata about the service from the service registry is retrieved using the service key.
- If the send PRI parameter is "Yes", collect all the necessary data required as per the portal request data specification and package that data into an XML document, which is added to the HTTP request header for transport to the service. (See APPENDIX E: PRI Data.)
- If a URI parameter did not come in on the request, get the URI to the service from the metadata (accessPoint). If a WSDL binding exists, get the URI of the WSDL file from the docLink attribute.
- Call the service with the PRIData (if applicable), passing through any service-specific parameters (cookies, form name/value pairs).

- Receive the response back from the service.
- Query the HTTP status code of the response. Ensure the SOAP fault is applicable.
- In case of an authentication request (HTTP status 401) as part of a HTTP Basic authentication challenge, the 401 status code and the "WWW-Authenticate" header will be passed to the user's browser. If used, authentication realms must be unique across all UFSs. To facilitate this, we recommend that the realm be set to the fully qualified DNS name of the UFS server. When the browser sends an "Authorization" header to the portal it will be transmitted in the request to the UFS. For NIPRNET security policy compliance, HTTP Basic authentication must be used only in conjunction with SSL. This process is unnecessary for UFSs that are SSO integrated. Other HTTP authentication schemes (i.e. Digest) are not compatible with the portal URL rewrite.
- In case of an authorization problem (HTTP status 403), a standard form for requesting access to the service will be sent back to the user. The standard form is prepopulated with service contact information from the service registration metadata.
- If another type of error is returned, it is passed on to the user.
- If content is returned, then the MIME type is examined. If the MIME type is text/XML and RenderXML=Y, the XML parser is called to transform the XML/XSL and return HTML.
- Run the HTML through the hyperlink re-write filter code, which prepends the URL of this portal to all identifiable URIs. If rewrite URL is set to No, this is skipped.
- Render the HTML content back to the portal for display.

Figure 16: Portal Response Processing illustrates the processing performed by the portal after the response is received from the UFS.

2.1.7.4.5 Portlet Service Response

The response of the portal is described fully in Table 6: Portal Response Processing. The portal manages the behavior of this interface.

2.1.7.4.6 Postconditions

Each invocation of the portal is stateless. The portal caches no information from previous invocations.

Table 6: Portal Response Processing

FaultCode	FaultString	FaultActor	Caller Response
InvalidParameter	An invalid parameter or value was detected: parameter, value	All	Service was not called.

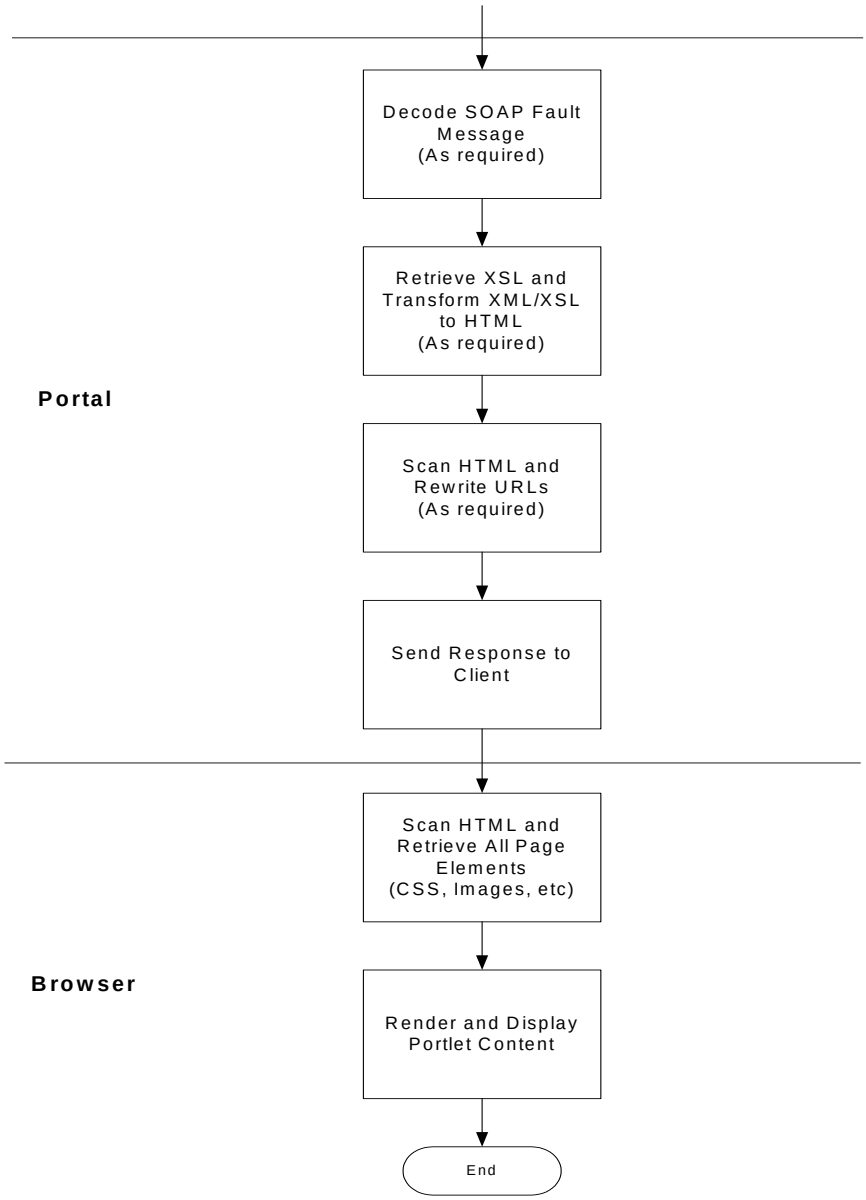


Figure 16: Portal Response Processing

2.1.8 User Facing Service Interface

The purpose of this section is to provide a concise, understandable description of the interface between the portal and a UFS. This interface is highlighted in Figure 17: User Facing Service Interface Section. An application developer creates a UFS when web content needs to be migrated to the NEP. The developer needs specific information to develop a UFS that will correctly communicate with the portal. This section provides the necessary information for the UFS developer.

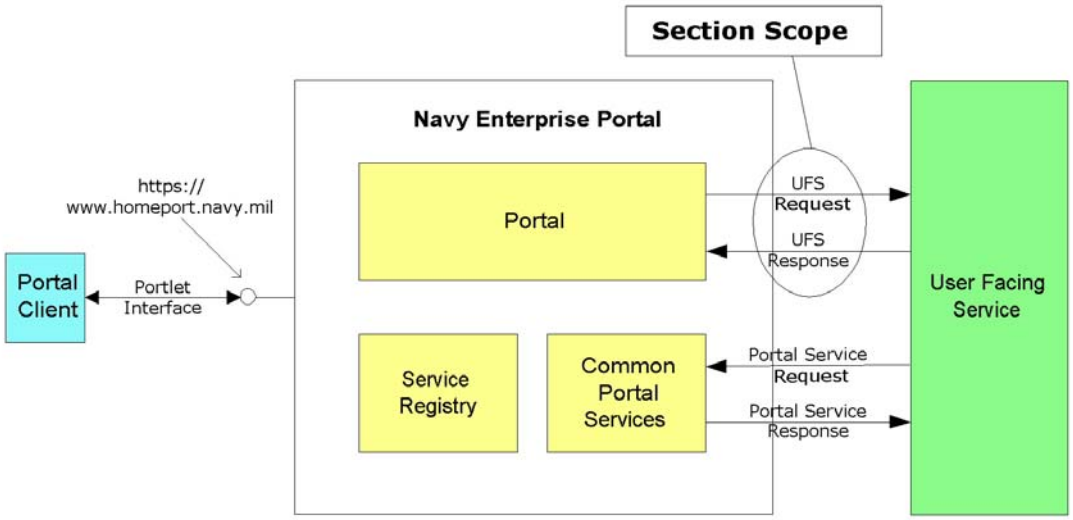


Figure 17: User Facing Service Interface Section Scope

The interface description is provided in the subsequent text and fully describes the UFS interface, but does not describe any context where the interface is used. Please see 2.1.5.2.4 NEP Execution Sequence for a discussion of all interactions within the NEP, including how and when the UFS interface is used.

2.1.8.1 Interface Documentation Approach

The approach used to describe the portal-UFS interface is frequently used, but may not be familiar to all readers. Therefore, a short explanation is provided in this section.

The description is composed of five parts:

- Preconditions define all of the requirements that must be satisfied before a UFS request can be sent from the portal to the UFS. These preconditions may include both real-time software execution conditions and policy conditions.

- UFS Request describes the different types of UFS requests that can be sent from the portal to the UFS. Content and formatting of the messages are also described, although lengthy descriptions are usually referenced instead of being included.

- UFS Responsibilities describes the tasks that need to be performed by the UFS. These tasks may differ depending on the type of message received and the values of fields within the message.

- UFS Response describes the different types of UFS responses that are acceptable to the portal. UFS developers must create UFS responses that meet the conditions in this section. Both content and format are described in this section.

- Postconditions define all the requirements that will be satisfied after the portal receives a UFS response.

The application developer should be familiar with all five parts of the interface description in order to create a valid UFS. Detailed information for each part is provided in the following sections.

2.1.8.2 UFS Interface Overview

UFSs are the software components that create and return content to the portal for display as a portlet. The portal sends a UFS request to the UFS, receives a UFS response, performs some processing as defined later in this section, and then sends the (possibly modified) content to the portal client.

At present, the UFS should only return content to the portal after a UFS request has been received. The portal will create and send a UFS request at various times under a variety of conditions. Most of the UFS requests are the result of a portal client action, e.g., requesting an update of the content display.

Development of the UFS is the responsibility of developers working for (or on behalf of) the back-end application providing content. If the determination is made that the appropriate level of integration is a UFS, defining the characteristics of the portlet interface will shape the way that the UFS is called.

2.1.8.3 UFS Interface Definition

2.1.8.3.1 Preconditions

The following preconditions must be satisfied before a UFS request is sent:

- The network path part of the URL (e.g., www.mysystem.navy.mil) provided to the NEP for addressing the UFS must be resolvable by a Domain Name Service (DNS) available to the portal.
- The path and filename of the system-dependent part of the URL (e.g., mydir/startpage.html) provided to the NEP must exist.
- The access point, a complete URL that includes both of the parts above, as well as the addressing scheme (e.g., *http://www.mysystem.navy.mil/mydir/startpage.html*), must be accessible by the portal.

2.1.8.3.2 UFS Request

UFS request messages are always transferred using HTTP or HTTPS. The format of the message always meets the definition of an HTTP request message as described in RFC 2616, Hypertext Transfer Protocol – HTTP/1.1 (see http://www.ietf.org/rfc/rfc2616.txt).

UFS request messages fall into two categories:

- Messages without a SOAP envelope
- Messages with a SOAP envelope

The portal determines the type of message to send based on metadata supplied when the UFS is submitted to the NEP. If the submitter of the UFS has indicated the UFS is a web service, then a message with a SOAP envelope is sent. Otherwise, a non-SOAP HTTP message is sent.

For both types of messages, the UFS request message may optionally include two data elements that are unique to the NEP:

- PRI data block may be included in the HTTP header or SOAP header depending on the message type. (See APPENDIX E: PRI Data.)

- The common identity of the user may be included in an HTTP header. If requested, the common identity will be part of the header in the HTTP message. (Normally, the web server on the platform hosting the UFS provides a method for accessing the header field values. Details vary by web server version and supplier, but this value is frequently made available to developers as the environment variable AUTH_USER.)

2.1.8.3.3 UFS Responsibilities

In general, the UFS is responsible for accepting the UFS request from the portal and generating a UFS response from which a portlet can be displayed. (The UFS response is described later in 2.1.8.3.4 UFS Response.) As long as the UFS addresses this basic responsibility, the developer is free to design and construct the UFS as he or she sees fit.

Certain activities are typical and common to all UFSs.

Figure 18: Typical UFS Activities is a flow diagram that describes these typical activities. The diagram is presented here with the hope of reducing the time and effort required for developing a UFS.

The responsibilities of the UFS with regard to security are fairly complex and dependent on the specific security policies in effect for the UFS domain. Please see APPENDIX G: Application Security for a discussion of the alternatives available to a developer for handling security issues.

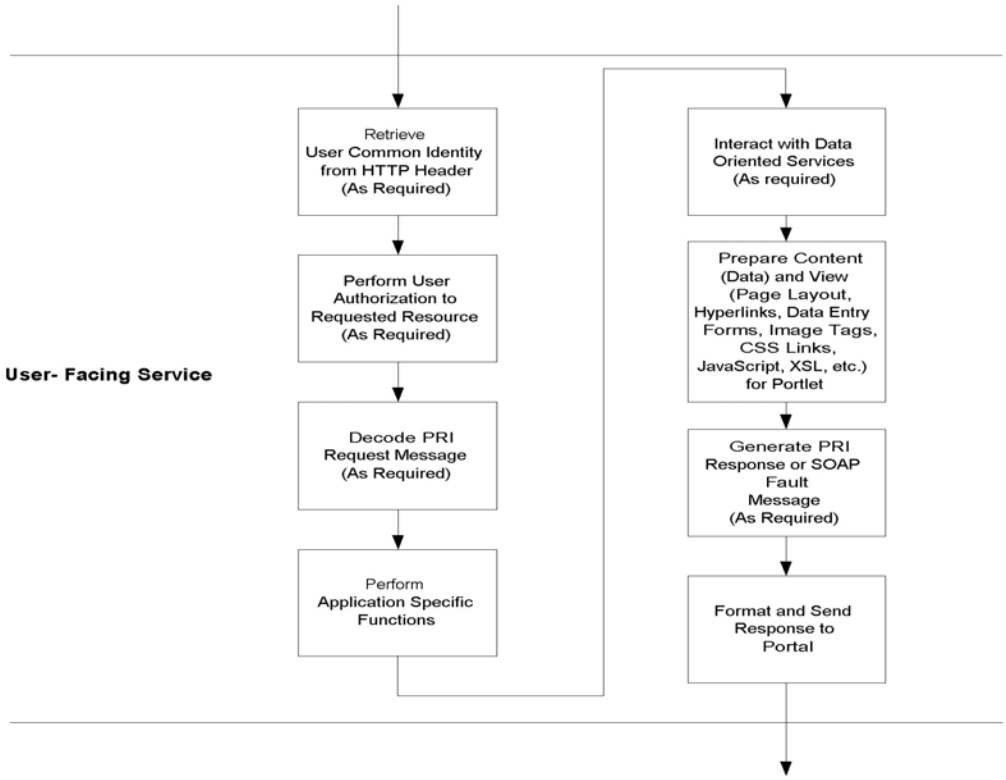


Figure 18: Typical UFS Activities

2.1.8.3.4 UFS Response

Four types of UFS responses are valid:

- A response without a SOAP envelope that reports a normal status
- A response without a SOAP envelope that reports an error status
- A response with a SOAP envelope that reports a normal status
- A response with a SOAP envelope that reports an error status (also referred to as a SOAP fault)

2.1.8.3.4.1 Normal UFS Responses

In most cases, a UFS response (non-SOAP or SOAP) will contain either of the following:

- HTML
- XML with an embedded XSL reference

Although most responses will contain one of the above, the NEP supports any response that complies with the HTTP 1.1 specification. For example, a Scalar Vector Graphics (SVG) file or a Portable Document Format (PDF) file could be returned by a UFS. For these special cases, the appropriate MIME type should be set in the HTTP header, and the developer should confirm that the client browser supports the MIME type being returned.

2.1.8.3.4.2 UFS Error Responses

Error reporting is handled differently for non-SOAP and SOAP responses. For non-SOAP error responses, utilize the standard HTTP 500 error reporting techniques. See RFC 2616, section 10.5 for more information.

A SOAP response after an error should be a standard SOAP fault message. The SOAP fault message should include an error code and error message.

2.1.8.3.5 Postconditions

If the UFS needs to maintain state for the client's portal session, then the UFS must save all necessary attributes of the state as well as the session ID that is provided in the PRI data included in the UFS request. Please note that current DoD policy allows the use of temporary session cookies with web browsers, but not persistent cookies.

2.1.9 Portal Services Interfaces

The purpose of this section is to provide a concise, understandable description of the interfaces to common services exposed by the NEP framework (See Figure 19: Portal Services Interface Section Scope). An application developer creates the service when content needs to be migrated to the NEP. A developer does not need to implement any of the services referenced here, but they are made available as portal services to allow for consistent implementation of the functions performed. If a developer chooses to implement an interface to one of the portal services, this section provides information describing the interface, its messages, and behavior.

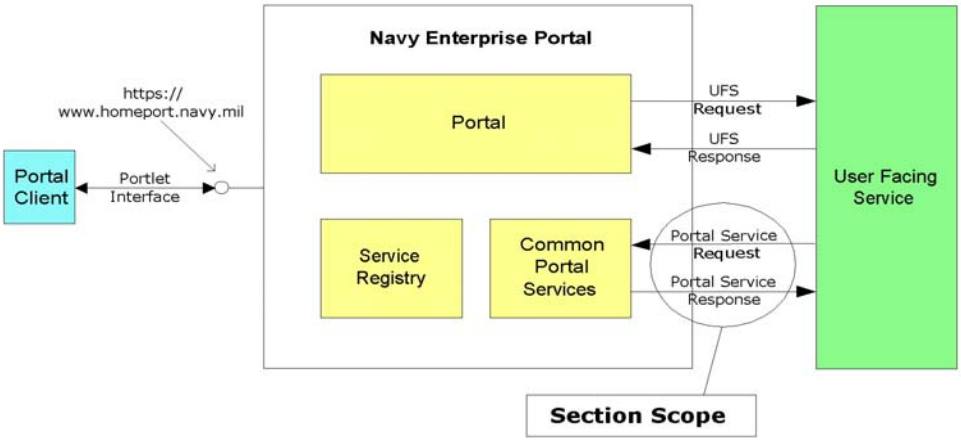


Figure 19: Portal Services Interface Section Scope

The interface descriptions provided in the following text include a description of all valid options. However, the appropriateness of one option vs. another option is not discussed.

2.1.9.1 Interface Documentation Approach

The approach used to describe the set of portal services interfaces is frequently used, but may not be familiar to all readers. Therefore, a short explanation is provided in this section.

The description is composed of five parts:

- Preconditions define all of the requirements that must be satisfied before a service request can be sent to the portal service. These preconditions may include both real-time software execution conditions and policy conditions.

- Portal Service Request describes the different types of service requests that can be sent to the portal service. Service developers must create service responses that meet the conditions in the section. Content and formatting of the messages are also described, although lengthy descriptions are usually referenced instead of being included below.

- Portal Service Responsibilities describe the tasks that need to be performed by the portal service. These tasks may differ depending on the type of message received and the values of fields within the message.

- Service Response describes the different types of service responses that are produced by the portal service. Both content and format are described in this section.

- Postconditions define all the requirements that can be assumed to be met after a portal-service call.

The application developer should be familiar with all five parts of the interface description in order to properly use the common service. Detailed information for each part is provided in the following sections.

2.1.9.2 Portal Services

The portal services interfaces published for use by UFS developers are described in the following sections and summarized in Table 7: Portal Services.

Table 7: Portal Services

Interface	Description	Section
Service Registry	Enable access to the enterprise service registry.	2.1.9.2.1:Service Registry Interface
URL Rewrite	Proxy calls to web sites through the portal.	2.1.9.2.2: URL Rewrite Interface
Web Service Execution	Expose common service for parameterized calls to web services.	2.1.9.2.3: Web Service Execution Engine (WSEE) Interface
XML Transformation	Expose the common XML rendering engine.	2.1.9.2.4: XML Transformation

2.1.9.2.1 Service Registry Interface

The NEP service registry web service interface is exposed so that NEP backend application developers can programmatically query the service registry. The interface exposed is the UDDI v2.0 standard inquiry API interface. As this is an industry-defined interface, it is not separately described here. For more information, please refer to the documentation located at http://www.uddi.org. The service registry interface is a private UDDI registry and is callable from the following URL from all NEP instances: https://portal.tfw.navy.mil/ServiceRegistry/webservices/uddiapi.asp (sample link, not active)

Since the Service Registry Interface is a SOAP-based web service, the following WSDL Document describes it and is evocable from the following URL from all NEP instances: https://portal.tfw.navy.mil/ServiceRegistry/webservices/uddiAPIWSDL.asp (sample link, not active)

The methods of the Service Registry Inquiry API can also be invoked via the WSEE Interface.

In addition, a web-based service registry browser application is available; see 2.1.11.3: Service Registry Browser for more information.

2.1.9.2.2 URL Rewrite Interface

The URL rewrite web service interface allows access to the features of the URL rewrite service. In general, the URL rewrite service will examine an HTML stream and prepend the portlet interface URL to each URL reference to allow the portal to proxy communication through the portal. A more complete discussion of the URL rewrite proxy rationale and restrictions is contained in APPENDIX H: URL Rewrite Guidelines. The URL rewrite service is callable from the following URL from all NEP instances: https://portal.tfw.navy.mil/nep/URLRewrite?<parameters> (sample link, not active)

The URL Rewrite Service can be invoked via a standard HTTP Request or a SOAP Request over HTTP. Since the URL Rewrite Service Interface is exposed as a SOAP-based web service, the following WSDL Document describes it and is evocable from the following URL from all NEP instances: https://portal.tfw.navy.mil/nep/CommonServicesWSDL (sample link, not active)

The URL Rewrite Service can also be indirectly invoked via the WSEE Interface.

The URL Rewrite Service supports three types of invocation:

- Direct invocation via a standard HTTP Request
- Direct invocation via a SOAP Request
- Indirect invocation via the WSEE Interface

When the URL Rewrite Service is invoked via a standard HTTP Request, the input parameters are gathered from the HTTP Request's Query String or Form Data. Below is an example of a URL Rewrite Service invocation via a standard HTTP Request:

https://portal.tfw.navy.mil/nep/URLRewrite?contentURL=http://www.myservice.mil/myapp.jsp

When the URL Rewrite Service is invoked via a SOAP Request, the input parameters are gathered from the SOAP Request message located in the body of the HTTP Request. When the URL Rewrite Service is indirectly invoked via the WSEE Interface, the input parameters are gathered from the WSEE Interface and the WSEE will dynamically read the WSDL, build the SOAP Client message and handle the invocation of the URL Rewrite Service.

2.1.9.2.2.1 Preconditions

The URL rewrite service parses HTML. There are restrictions and conditions that HTML must meet to be successfully parsed by the service. In general, the URL references in the stream passed to the rewrite service must be identified as valid URLs by the parser. URL fragments or URLs generated on the client are generally not supported. For more information, please refer to APPENDIX H: URL Rewrite Guidelines.

2.1.9.2.2.2 URL Rewrite Request

Table 8: URL Rewrite Request

Parameter	Values	Description	Required
Action	n/a	The ACTION parameter is optional and has not been implemented at this time.	N
contentURL	URI	A valid URL to the HTML content that will be run through the URL Rewrite proxy engine. The URL Rewrite Service will only proxy valid HTML content.	Y
baseURL	[URI]	For relative links, a baseURL can optionally be provided for inclusion in the rewritten URL.	N

The URL rewrite session cookie is named *URLRewriteSession* and contains a string that is passed as the sessionParameters parameter to the service. In cases where the cookie was not produced, a

rewritten URL contains the same string located between the slash following the portal client interface URL https://portal.tfw.navy.mil/nep/PortalConnector/ (sample link, not active) and a special character (@) used to identify the end of the rewrite parameters.

2.1.9.2.2.3 URL Rewrite Responsibilities

The URL rewrite service will perform the following actions:

- Parse the stream passed
- Identify URL references in the stream, applying the baseURL as required
- Prepend the session state to the target URL
- Prepend the portlet interface URL to the target URL
- Return the stream to the caller

A rewritten URL takes the following format:

https://portal.tfw.navy.mil/nep/PortalConnector/[sessionParameter]@[targetURL]] (sample link, not active). More detailed information on processing, restrictions, and constraints of the URL rewrite service can be found in APPENDIX H: URL Rewrite Guidelines.

2.1.9.2.2.4 URL Rewrite Response

Table 9: URL Rewrite Response

Message	Message Content	Description	Required
Stream	String	The rewritten URL or HTML stream.	Y
Fault	XML	SOAP faults are returned if errors are encountered.	N

2.1.9.2.2.5 Postconditions

All URL references passed will be rewritten if they can be identified as URL references by the service. The session state is saved in either a session cookie or in the URL.

Table 10: URL Rewrite Postconditions

FaultCode	FaultString	FaultActor	Caller Response
InvalidParameter	An invalid parameter or value was detected: parameter, value	All	Service was not called.
BadSessionParameter	The sessionParameter value was missing or invalid	All	Service was not called. No rewriting was performed.

2.1.9.2.3 Web Service Execution Engine (WSEE) Interface

The WSEE is a facility within the portal that supports parameterized execution of industry standard web services. Most commercially available web-services execution engines will support calling a web service based on a WSDL file. The NEP WSEE allows the operation and parameters to be supplied so

that additional interaction with the user to establish the parameters of service execution can be automated. As web-services standards evolve, it is assumed that standards such as the OASIS Web Services for Remote Portal (WSRP) protocols will formalize the methodology for doing this in a language-independent manner.

The WSEE Interface is exposed as an HTTP Request interface to the Portal Application and the Portal Client that allows dynamic execution of SOAP based Web Services via a standard HTTP Request. Conceptually, the WSEE Interface links the Portal Application and the Portal Client to the cloud of WSDL-Bound User Facing and Data-Oriented Services exposed to the Navy Enterprise Portal. The WSEE Interface is used to dynamically invoke Web Services that are evocable via SOAP 1.1 messaging, whose interfaces are described by WSDL 1.1, to which can a reference can be published and locatable in a UDDI 2.0 registry. The WSEE Interface is implemented in the Portal Connector GOTS software component and is evocable from the following URL:

https://portal.tfw.navy.mil/nep/WSExecute?<parameters> (sample link, not active)

The parameters that comprise the WSEE Interface are designed to minimize the complexity of invoking a SOAP based Web Service. The WSEE Interface input parameters can be supplied in three ways:

- As a set of query string parameters
- As a set of HTTP Post Form Data parameters or
- As a mixture of query string and HTTP Post parameters

Each of the WSEE parameters are reserved names and should not be used by applications for other purposes.

2.1.9.2.3.1 Preconditions

- The service has to be callable as a standard web service.
- The service interface must be defined in a WSDL file.
- The WSDL file must be registered in the NEP enterprise service registry or a public UDDI registry with a WSDL tModel.

2.1.9.2.3.2 WSEE Request

Table 11: WSEE Request

Parameter	Values	Description	Required/ Reserved
uddiBindingKey	[GUID]	The UDDI bindingKey of the service to be executed.	Y/Y
operation	[operation-name]	The operation name as defined in the WSDL file of the operation to be executed. If no operation is specified, a page will be generated listing the operations that a user can select from.	N/Y

Parameter	Values	Description	Required/ Reserved
operationMessageParts	[MessagePartName1~MessagePartValue1#MessagePartName2<MessagePartValue2] Note: The "~" character is used to delimit MessagePartNames and MessagePartValues. The "#" character is used to delimit the pairs of MessagePartNames and MessagePartValues	The message parts for the input message to the service's operation as defined in the WSDL file. The keyword MessagePartName will be validated against the message part names in the WSDL file. The MessagePartValue will be passed as the parameter value to the service's operation. For any required input message parts for which a value is not specified, a page will be generated requesting the information from the user.	N/Y
uddiURL	[URI]	The default behavior of the WSEE Interface is to lookup all uddiBindingKey references in the private NEP Service Registry. If the service is defined in a public UDDI registry, the Uniform Resource Identifier (URI) to the Inquiry API Interface must be specified here.	N/Y

Special Notes on OperationMessageParts: The OperationMessageParts can be supplied as a set of HTTP Post Form Data name\value pairs. This feature allows clients to issue HTTP Posts with Operation Message Part name\value pairs passed via the HTTP Post Form Data to the WSEE Interface without having to pre-format the OperationMessageParts input parameter. The limitation to this feature is that the WSEE Interface must employ the above *reserved parameter names*. If a particular Web Service Operation has an input Message Part whose name is equivalent to one of the WSEE Interface's reserved parameter names, then a client cannot pass the Message Part and value as an HTTP Post Form Data name\value pair. In the cases where there is a naming conflict with a WSEE Interface reserved parameter and an Operation's Message Part, the client will need to pre-format the OperationMessageParts input parameter per the above design before invoking the WSEE Interface. Once correctly pre-formatted, the OperationMessageParts input parameter\values can be passed to the WSEE Interface via either the HTTP Post Form Data or across the HTTP Request Query String.

2.1.9.2.3.3 WSEE Interface Examples

The following WSEE Interface instance is invoked with all of the input parameters passed on the HTTP Request query string. In this example, the WSEE Implementation will default to querying the private NEP Service Registry:

```
https://portal.tfw.navy.mil/nep/WSExecute?uddiBindingKey =C2B6F320-XBC4-4635-1415-
D01671B7C649&operation=getStockQuote&operationMessageParts=tickerSymbol~MSFT#closingDate~
11/05/2000
```

The following WSEE Interface instance shows how to query a public UDDI 2.0 Registry whose Inquiry API address is specified with the uddiURL query string parameter:

```
https://portal.tfw.navy.mil/nep/WSExecute?uddiBindingKey         =C2B6F320-XBC4-4635-1415-
D01671B7C649&operation=getStockQuote&operationMessageParts=tickerSymbol~MSFT#closingDate~
11/05/2000&uddiURL=http://SomePublicUDDIServer/Inquiry
```

The following WSEE Interface instance is invoked using a standard HTML Form, which will issue an HTTP Post to the WSEE Interface. The input parameters are passed to the WSEE Interface via HTTP Post Form Data name\value pairs. Upon submittal of the HTML Form, the browser will generate an HTTP Post to the WSEE Interface (see key data elements highlighted in red). In this example, the WSEE Implementation will again default to querying the private NEP Service Registry.

```
<HTML>
<HEAD>
        <TITLE>WSEE Interface HTTP Post Form Data example</TITLE>
</HEAD>
<BODY><P><BR></P>
<H3>A HTML Form POST invocation of the WSEE Interface that will dynamically execute a
getStockQuote operation</H3>
<FORM      name="frmgetStockQuote"      action="https://portal.tfw.navy.mil/nep/WSExecute"
method="post">
        <DIV style="DISPLAY: none">
                <INPUT                          value="78B2CCC9-D2C4-44A7-9602-18915E0B9811"
                name=" uddiBindingKey">
        </DIV>
        <TABLE>
        <TR><TH>getStockQuote   Web   Service   operation   invocation   via   the   WSEE
Interface</TH></TR>
        <TR><TD>Ticker Symbol Message Part</TD><TD>
                <INPUT value="getStockQuote " name="operation">
                <INPUT size="15" value="MSFT" name=" tickerSymbol">
                <INPUT size="15" value="11/05/2000" name=" closingDate">
        </TD></TR>
        <TR><TD><INPUT id="submit1" type="submit" value="Invoke" name="submit1"></TD></TR>
        </TABLE>
</FORM>
<P><BR></P></BODY>
</HTML>
```

2.1.9.2.3.4 WSEE Responsibilities

The WSEE will retrieve the WSDL file from the UDDI registry.

The WSEE will format a call to the service operation selected using the parameters supplied over an HTTP binding.

The WSEE will return the result to the portal for processing.

2.1.9.2.3.5 WSEE Response

Table 12: WSEE Response

Message	Message Content	Description	Required
Stream	String	The result of service execution. This is the operation output message as defined in the WSDL file.	N
Fault	XML	SOAP faults are returned if errors are encountered.	Y

2.1.9.2.3.6 Postconditions

Table 13: WSEE Postconditions

FaultCode	FaultString	FaultActor	Caller Response
InvalidParameter	An invalid parameter or value was detected: parameter, value.	All	Service was not called.
BindingKeyNotFound	The requested bindingKey could not be found in the registry specified.	All	The service could not be executed. Validate that the correct bindingKey was specified.
OperationNotFound	The operation specified could not be located in the WSDL file.	All	The operation could not be executed. Validate that the correct operation was specified.
MessagePartNotFound	The specified messagePart could not be located in the WSDL file.	All	The message part could not be found. No action was taken with the specified value. The service was executed but the returned information may not be valid.

2.1.9.2.4 XML Transformation Request

The XML transformation web service provides a common implementation of XML rendering using XSLT style sheets. It has been observed that many of the implementations of XSLT rendering services can generate different results from the same source XML and XSLT files. Use of a common service allows a consistent result. The XML transformation service is callable from the following URL from all NEP instances: https://portal.tfw.navy.mil/nep/XMLTransform?<parameters> (sample link, not active)

The XML Transform Service can be invoked directly via a standard HTTP Request or a SOAP Request over HTTP. Since the XML Transform Service Interface is exposed as a SOAP-based web service, the following WSDL Document describes it and is available from the following URL from all NEP instances: https://portal.tfw.navy.mil/nep/CommonServicesWSDL

The XML Transform Service supports three types of invocation:

- Direct invocation via a standard HTTP Request
- Direct invocation via a SOAP Request
- Indirect invocation via the WSEE Interface

When the XML Transform Service is invoked via a standard HTTP Request, the input parameters are gathered from the HTTP Request's Query String or Form Data. Below is an example of a XML Transform Service invocation via a standard HTTP Request:

https://portal.tfw.navy.mil/nep/XMLTransform?xmlContent=http://www.myservice.mil/myXMLDoc.xml&xsltURL=http://www.myservice.mil/myXSLDoc.xsl

When the XML Transform Service is invoked via a SOAP Request, the input parameters are gathered from the SOAP Request message located in the body of the HTTP Request. When the XML Transform Service is indirectly invoked via the WSEE Interface, the input parameters are gathered from the WSEE Interface and the WSEE will dynamically read the WSDL, build the SOAP Client message and handle the invocation of the XML Transform Service.

2.1.9.2.4.1 Preconditions

- XML must be well formed.
- XML is not checked for validity.
- An XSL file must be referenced as an absolute URI with a fully qualified domain name (http://app.navy.mil/file.xsl)
- An XSL file reference URI must be either
- Passed by the caller
- Embedded by reference in the XML document

The XSLT URI reference is assumed to be included in the stream.

2.1.9.2.4.2 XML Transformation Request

Table 14: XML Transformation Request

Parameter	Values	Description	Required
Action	[renderXML]	The ACTION parameter controls the type of processing performed by the service. Only one action is currently supported. The parameter is supported for extensibility purposes.	N
xmlContent	URI or raw XML Content	The xmlContent argument can contain a URI to a valid XML Document or raw XML text. If the argument is a URI, the Web Server serving the XML document needs to set the Content Mime Type or associate the .xml extension to the 'text/xml' Mime Type.	Y
XsltURL	<URI>	The xsltURL argument should be an absoluteURI to a XSLT Document that will be used to transform the XML Content gathered from the xmlContent argument. If provided, the XML Content will be rendered using this Stylesheet. If not provided, the XSLT URL is assumed to be declared in the XML Content. In either case, the XSLT URL must be a fully qualified URL to the XSLT Stylesheet. The Web Server serving the XML document also needs to set the Content Mime Type or associate the .xsl extension to the 'text/xml' Mime Type.	N

2.1.9.2.4.3 XML Transformation Responsibilities

The XML and XSLT (if provided) are passed to the XSLT parser.

The result is returned to the caller.

2.1.9.2.4.4 XML Transformation Response

Table 15: XML Transformation Response

Message	Message Content	Description	Required
Stream	String	The rendered XML or HTML	N
Fault	XML	SOAP faults are returned if errors are encountered.	Y

2.1.9.2.4.5 Postconditions

If no XSLT reference is found, the XML stream is returned unchanged and a fault is returned.

Table 16: XML Transformation Postcondition

FaultCode	FaultString	FaultActor	Caller Response
InvalidParameter	An invalid parameter or value was detected: parameter, value.	All	Service was not called.
NoXSLTReference	No XSLT reference was identified. No transformation was performed.	All	Input is passed through to the output stream unchanged.

2.1.9.3 Summary

The services exposed by the portal allow a UFS developer to further customize the interface between the portal and the UFS through the use of callbacks to the portal. This functionality is intended for customization in cases where the standard functionality of the portal cannot satisfy the needs of the backend application service. It is likely that there will be an impact to the UFS over time as the NEP portal framework evolves and deprecates these application program interfaces (APIs) in favor of evolving standards such as Web Services for Remote Portal (WSRP), WSIA (Oasis), the JSR 168 Java portlet standard, and other possibilities not yet established enough for implementation support.

2.1.10 Infrastructure and Design Considerations

To protect NMCI systems from the threat of malicious or improper use of mobile code, developers must assess and control the risks imposed by the technology. The guidance provided in the DoD Mobile Code Policy should be the first step in an iterative process to reduce such risks to NMCI information systems. It categorizes mobile code technologies and restricts their application based on their potential to cause damage if used maliciously. Developers should consult and follow the guidance provided by the DoD Mobile Code Policy, which can be found on the link provided in APPENDIX B: References.

2.1.10.1 Navy Marine Corps Intranet

NMCI implements the applicable policies described above. Many of these policies will affect the developer when migrating applications and services to the NEP. In addition, there are best practices

and considerations that should be considered when interacting with the NMCI environment. Some of these considerations are described in the following text.

2.1.10.1.1 NMCI Client Seats

The NMCI client seats are managed seats. As such, a user does not have administrative access to modify the configuration or install specialized software. The base build is installed as part of a client gold disk and contains the common tools and programs available for use. Additional components can be installed on a client, but these are pushed to the desktop by an administrative facility. As a user cannot add software to his or her seat, it is important for content developers to verify that client plug-ins and components are available and compatible. The gold disk contents can be found at http://www.nmci-isf.com.

If a dependency on a desktop application (such as Adobe Acrobat Reader) exists and the application is not currently on the NMCI desktop gold disk, then a request must be made for the application to be included on the NMCI desktop according to the standard NMCI certification process. Software on the desktop can only write to the "My Documents" folder. Any client-side components should be certified as Windows 2000 compliant and should not attempt to write to restricted areas of the client.

2.1.10.1.2 Designing For Performance

In a highly distributed web application such as the NEP, it is important for developers to consider the impact of design choices on application performance. While the NMCI network is a high-bandwidth network, some of the users of enterprise applications may be accessing the service from afloat platforms or from other locations with limited bandwidth availability. Performance is restricted to the lowest common denominator when it comes to network performance. The slowest segment between the user and the service governs latency and impacts response time.

There are several ways in which a developer can impact performance. The NMCI web services infrastructure supports caching, compression, content distribution, and high-availability hosting. Also, it is good practice for service developers to use scanning and packet monitoring techniques to understand the number of "round trips," "chattiness," and bandwidth requirements of the service being integrated and design for optimal efficiency.

2.1.10.1.2.1 Content Caching

Under HTTP, each HTML page rendered by the portal makes a separate HTTP GET request for each file or graphic referenced in the document. The user experience can be significantly improved by allowing non-dynamic content to be cached. Care should be taken to restrict the use of content expiration or no-cache directives to dynamic content. This improves network throughput and benefits all users of the infrastructure.

2.1.10.1.2.2 Content Distribution

The content cache servers can be configured to distribute content on a scheduled basis, pushing that content out to the "edge" closer to the user. This is beneficial in that static content is referenced without traveling all the way back to the application server. Only the dynamic content needs to be served from the source server.

2.1.10.1.2.3 Compression

Most web servers and clients support HTTP compression. Web servers can be configured to enable compression and will only compress content if the client indicates that it can process the response. In addition, it may be advantageous to reduce round trips by using a multi-part MIME or SOAP with attachments to return all of the files for a content pane at once, allowing the client to recombine the result.

2.1.10.1.2.4 High Availability or Mission-Critical Hosting

Web content hosted by NMCI can be deployed for mission-critical, high-availability performance. This involves placing content at multiple hosting facilities and using a global load-balancing solution to route user requests to the nearest available instance hosting that content. Web hosting on NMCI may be accomplished via CLIN 029 "Legacy Systems Support" other standard co-loction hosting solutions have not been priced or added to the NMCI service catalog as of this writing.

2.1.10.1.2.5 Other Considerations

Use of network monitoring tools can help a developer to design his or her service for optimal performance. As previously discussed, the number of round trips per page can impact the total time to render the content to a user. A high number of graphics or embedded files must be transferred individually, and the latency of each transfer must be added up. Large graphics should be optimized for size. A developer should also look for "chatty" network conversations. Once again, many trips multiplied by latency can negatively impact perceived performance of the service.

Interface design can impact the chattiness of an interaction. As systems are developed, normally many public and private methods or subroutines are developed. It is generally considered poor practice to expose all of the individual methods. An interface that encapsulates the business rules would be preferable, so that a user can make fewer calls to get the data and ensure that the business rules are applied.

2.1.10.1.3 Security Policy and PKI 3.0 Requirement

One of the most common issues that prevents integration of content into the portal or into NMCI in general is related to compliance with security and firewall policy. In general, use of HTTP(s) protocols will allow interoperability and firewall compliance. Note that the NEP at https://portal.tfw.navy.mil is DoD and DON security-policy compliant using two-way SSL and requires the installation of a DoD Class 3 PKI identity certificate. Users must contact their local certificate authority to obtain their PKI certificate.

DON components shall continue to aggressively implement the DoD PKI in concert with adopting the Common Access Card (CAC), in accordance with reference (A), (B), (C), and (D). PKI provides digital identification, signature, and encryption services to a broad range of applications at various levels of assurance. PKI is an enabling technology that will reduce access management administration while increasing overall security and access control. Whenever appropriate, private rather than public web servers shall be utilized to further minimize exposure and enhance operational security by limiting data aggregation opportunities. All private web servers shall be issued DoD PKI server certificates and shall use the certificates for server authentication via the Secure Sockets Layer (SSL) protocol. In addition, all DON information systems shall display the appropriate official DoD notifications for web site privacy and security in accordance with reference (F). All DON web site content shall adhere to reference (E).

Table 17: PKI References

Ref.	Source
(A)	DoDD O-8530.1 of 8 Jan 01 Computer Network Defense
(B)	DoDIO-8530.2 of 9 Mar 01 Support to Computer Network Defense
(C)	DEPSECDEF Memorandum
(D)	"Smart Card Adoption and Implementation" of 10 Nov 99
(E)	ASD C3I DoD Web Site Administration Policies and Procedures of 25 Nov 98
(F)	SECNAVINST 5720.47 of 7 Jan 99, DON Policy for Content of Publicly Accessible World Wide Web Sites

2.1.10.2 IT-21

The IT-21 NEP environment encompasses the Fleet NOC and each ship in the Navy. These environments impose unique challenges and limitations that developers need to be aware of. The portal users in these environments will be using locally installed portals to access both local and remote (reach-back) applications.

2.1.10.2.1 IT-21 Configuration Management (CM) Policy

In accordance with IT-21 CM policy, each application owner will utilize the Preferred Products List (PPL), System/Subsystem Interface List (SSIL), and Qualified Parts List (QPL) processes for adding government-off-the-shelf/commercial-off-the-shelf (GOTS/COTS) products to the PPL, QPL, and SSIL. More information about this process is available at https://jdms.spawar.navy.mil.

2.1.10.2.2 Afloat-to-Ashore Connectivity

Afloat-to-ashore connectivity often suffers from severe bandwidth limitations and occasional outages. Developers of shore-based, reach-back applications need to understand this restriction. The *Transport* element of the PRI data message will assist the application in detecting reach-back connections. Special attention should be paid to minimizing overall data transmission sizes. One consideration for all developers would be to provide for the selection of different XSL style sheets on a per-connection basis to provide lightweight versions of shore-based applications to ship-based users. This may include the use of smaller images, or removal of unnecessary images.

2.1.10.2.3 Shipboard Proxies

Each ship provides web proxies that require user authentication. Often, some of the local shipboard users are not on the approved list of users to connect to off-ship web sites. This list may be designated by rank. This often varies by ship, ship type, or operational requirements. Application developers need to be aware of this limitation and design their applications accordingly.

The proxies also do not currently provide caching to HTTPS (SSL) traffic. Therefore, any remote (reach-back) applications that use HTTPS will not benefit from caching. Developers of these applications should attempt to minimize page download sizes in order to improve page load times for the users. Bulky, oversized graphics are often a significant problem that can be easily corrected.

2.1.10.2.4 Ship's Configuration

Each ship, or ship type, may have a different set of on-board applications to perform its mission. Each on-board application may be at different release levels for each ship. Each shipboard portal will be configured to synchronize the service registry with the shore portals on a periodic basis. Developers need to be aware of the implications of this synchronization and the overall system configuration. Each developer with a UFS that has any dependencies on specific ships, or ship types, will need to provide a list of designated ships or ship types. This may even require the development of multiple UFSs to support these variations.

2.1.10.2.5 NEP IT-21 Portal Rollout

The NEP portal is following a rolling installation plan for all ships. Shipboard application developers will need to be aware of the rollout timeline in relation to their own product releases. This impacts existing shipboard applications on ships that have a newly installed portal. Those applications may not become available to portal users until they receive any required upgrades to support the portal.

2.1.11 Development Resources

This section lists some tools that are available that may help a developer integrate a service with the NEP. These tools are useful, but are not required.

2.1.11.1 NEP Open Source Site

An open source site (OpSS) has been established as the common source of all NEP development news and information. Developers are encouraged to use this site as their first stop on their integration path with the NEP and to stay updated on NEP information. Developers can also use this site to ask questions and share experiences with others. This site is located at https://tfw-opensource.spawar.navy.mil.

2.1.11.2 Test Portal Systems

Test portal systems have been established for developers to use during their initial development and testing. These systems are for test and development only. The configuration of these systems matches the production portal systems, but information will not be synchronized between the systems. The service registry on these systems is for testing purposes only. Please refer to the OpSS for information on how to register for access to the test portals. The benefit of these systems to the developer is being better able to submit fully integrated and tested UFSs.

2.1.11.3 Service Registry Browser

Developers will be able to browse the production service registry using a provided web-based service registry browser application in addition to the Registry UDDI 2.0 standard inquiry API that is provided (see 2.1.9.2.1:Service Registry Interface). This will allow developers to discover the services available across the enterprise and to establish contact with the owners of other services. The registry browser application is available on the production portal at https://portal.tfw.navy.mil.

2.1.11.4 Development Tools

There are many software tools available to assist with the implementation of services for the portal. Developers should consider using commercial tools to develop HTML, CSS, XML, XSL, WSDL, SOAP, and others.

2.1.11.5 Web Validation Tools

The following validation tools have been made available by the W3C at http://www.w3c.org:

- Validate HTML files with W3C
- Validate CSS files with W3C
- Validate XHTML files with W3C

A Section 508 validation tool is available from the Center for Applied Special Technology (CAST) at http://www.cast.org.

2.2 NMCI INTEGRATION

The Navy Program Management Office (PMO), in concert with the Information Strike Force, has developed the supplement A to this document, the NMCI Release Development and Deployment Guide (NRDDG), to provide detailed information and guidance to developers interested in migrating content, introducing new applications, or changing existing applications within NMCI. The guide is a consolidated source of information, guidance, and direction to developers who build or modify applications and/or to the acquirers of applications intended for use specifically within NMCI. The NRDDG is currently unavailable for download. Please contact the NMCI PMO to request a copy.

The "Design and Development" section of Supplement A (NRDDG) has up to date information pertatining to NMCI integration best practices. The NRDDG describes the interface points that "thick client" application developers may need to know to properly code, configure and package their application to work well on the NMCI. Software and programming best practices, NMCI specific application interface, and as test and certification lab dos and don'ts are detailed in this section. Network interface speces and boundary protection as well as other Navy and DOD policy are examined in this same section. Testing avenues and other considerations are also explored to help ensure that applications being introduced into the NMCI will pass certification and will ultimately work in the NMCI live environment.

This section seeks to identify infrastructure interfaces, APIs, and specifications for the various types of applications that will share the NMCI/IT-21 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/IT-21-aware applications that will work securely and harmoniously with common network resources. Both NMCI and TFWeb participate in this Object model via Active Directory (AD). A simple application justification for the certification matrix that may help the developer properly interface an application with NMCI is listed in **Error! Reference source not found. Error! Reference source not found.**.

Excellent resources that define these specifications are the "Windows Logo Program" that may be found on the Microsoft's developer network (MSDN) web site at http://msdn.microsoft.com/certification/download.asp, the Microsoft Platform Software Developer Kit (SDK) that documents the Win32 API, and Microsoft's Active Directory Service Interface (ADSI) model.

2.2.1 Boundary/Network Interface Specifications

Boundary protections are the standard sets of protections that define the interfaces within NMCI and between NMCI and other networks (see Figure 20: Network Boundaries). Boundary protections

enforce the policies required to connect to those external networks, provide security mechanisms for secure access to applications, and protect communities of interest (COIs) residing within NMCI. The type and strength of each security component is dependent upon the information protection requirements for a particular system. This is especially true for boundaries 1, 2, and 3 as labeled in Figure 18. Boundary 1 reflects the Navy Marine Corps Enclave Protection Policy. Boundaries 2 and 3 security mechanisms are flexible enough to meet the security requirements of various scenarios. Specific configuration parameters of the security components deployed at the various boundary levels are tailored to provide the level of protection necessary to protect the confidentiality, integrity, availability, accountability, and non-repudiation of NMCI.

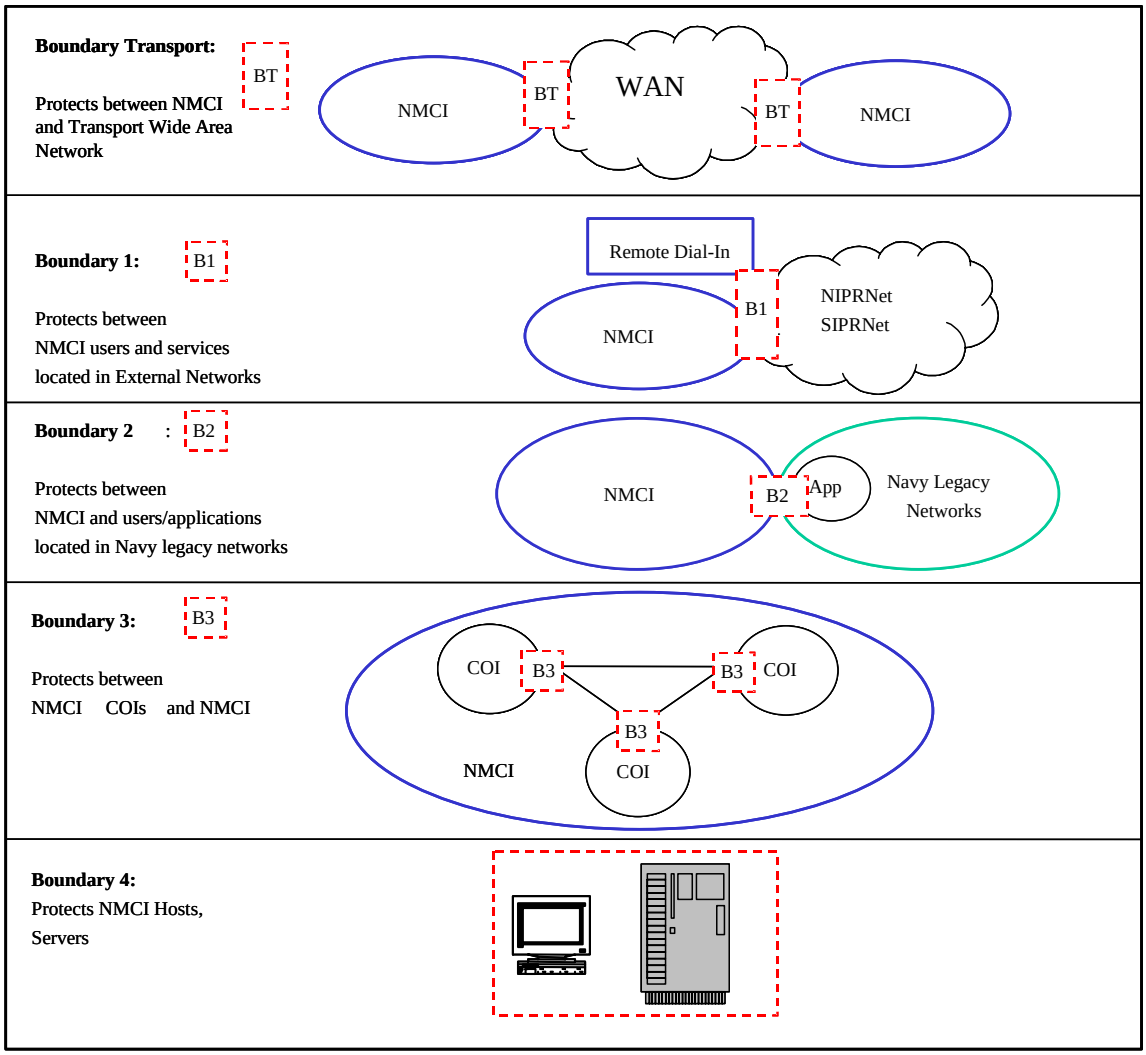


Figure 20: Network Boundaries

NMCI also provides a wide-area Internet protocol (IP) backbone using Defense Information Systems Agency (DISA) WAN services with Very High Speed Backbone Network Service (VBNS+) transport services. The transport boundary offers a secure encrypted path between bases while imposing minimal restrictions on inter-base communications.

2.2.1.1 NMCI Standard Browser

The official standard browser for NMCI at the time of this writing is Internet Explorer 5.5. Netscape 4.7 is also available to users and is also included in the Gold Disk standard image (see http://www.nmci-isf.com/downloads/Gold_disk_contents.pdf for the latest Gold Disk contents). Web development may be impacted because different browsers and browser versions do not support all web standards and may display content differently. Testing web applications under the standard NMCI browser is crucial.

2.2.1.2 Client/Server Network Sensitive and Mobile Code

Applications that require network connectivity for standard operation may, for the purposes of this document, be defined as "network sensitive." These applications must respect bandwidth rules to guarantee quality of service (QoS) to network resources.

2.2.1.3 Legacy Web Applications

The purpose of connecting NMCI and legacy networks is to allow users to continue to access systems or applications outside NMCI and to allow necessary interaction between components of systems that have elements on both NMCI and legacy networks. The ISF, by default, leaves application servers where they are during transition to NMCI. Each system or application uses protocols to communicate between clients and servers. Many protocols and ports are associated with security vulnerabilities, and boundary policy reflects this. If an external application is compliant with Boundary 1 firewall policy, then users within NMCI access through the boundary. To know if an application or system is compliant, its protocols, ports, and directions of activity must first be identified and characterized for comparison to those of NMCI.

If an external system requires interaction not allowed by Navy/Marine Corps firewall policy, there are technical methods to obtain access through the boundary. The Navy/Marine Corps may choose to make a modification to the baseline firewall policy to permit access to a system. Access may be possible through a Virtual Private Network (VPN) path. A risk assessment must be prepared to determine whether a modification to firewall policy or use of a VPN is acceptable. The NMCI Designated Approval Authority (DAA) and local DAA will use certification and accreditation (C&A) documents to assess risks and make firewall policy modifications. A risk assessment does not need to be a one-at-a-time process: several applications can be considered simultaneously, if they run on shared servers or use the same protocols.

The factors that influence risk assessment are as follows:

- Need for the particular protocol to traverse a boundary. For example, preliminary analysis of applications identified a number of applications and protocols that are associated only with the administration of particular systems and could be restricted to the legacy network if the administrators of those systems retain access to the legacy network.
- Vulnerabilities to NMCI associated with the use of protocol in question. These are captured in a risk model to be evaluated by the DAA.
- Ability to restrict use of the protocol in question to a finite address or address list. If the servers that run an application are static on the legacy network (rather than beyond on yet another network) and have fixed IP addresses, it is technically possible to filter packets and allow only those servers to pass an otherwise forbidden protocol.
- Security posture of the legacy servers in question (are the systems accredited?; do they have a security policy that controls and audits access and configuration?; do any security products protect them?). The accreditation package must cover the system and should follow Defense Information

Technology Security Certification and Accreditation Process (DITSCAP) or provide equivalent detail on configurations, risk, and security processes.

- Overall security posture of the legacy network, as expressed by its security accreditation status, security policy, and interfaces to additional external networks.
- Distribution and number of users of the subject application within NMCI.

No modifications to NMCI boundary policies are allowed without the NMCI DAA approval. When Marine Corps sites cut over, the involvement of both DAAs becomes critical, as there will be connections between NMCI and legacy networks and between Navy and Marine Corps COIs that need to be analyzed to determine mutually acceptable risk. If the risk of an exception to the boundary policy or use of a VPN is unacceptable, several options provide continued access to a legacy application.

- An alternative protocol or port. Usually not an option, but a potential for legacy applications that could be web enabled through an administrative change.
- Proxied access. For example, using a combination of web server and terminal server on the Demilitarized Zone (DMZ) to access multiple legacy applications. This approach effectively converts the offending protocol and allows legacy access to be arbitrated using PKI-based authentication. It also allows access to be closely monitored.
- Improving the security posture of the legacy server. This would include upgraded security policies, consolidating servers behind a protected subnet, using static IP to allow filtering, or moving to a DMZ off the NMCI boundary. Improvements must account for the trust granted the legacy server and the vulnerabilities associated with the protocols and services employed.
- Improving the security posture of the legacy network (this would include securing and consolidating its outer boundaries and upgrading and enforcing internal security policies). Enforcing Boundary 1 policies between a legacy network and its own external networks (e.g., Smartlink) does carry compatibility issues, because many legacy systems and their users span multiple sites. On the other hand, a single Boundary 1 suite could protect multiple external connections at a site.
- Use of two-sided VPN. This capability is built into the NMCI boundary design, but requires an extra VPN server in front of each legacy system. The advantage over a single-sided VPN is that the tunnel is extended into the legacy network, providing confidentiality and limiting access from other segments within the legacy network. This is appropriate where the legacy servers themselves pose low risk or can easily be protected.
- Kiosk computers (attached to legacy network) for continued access. This approach should be reserved to access applications that cannot be mitigated through other means.

All these options assume that an application needs to remain in service, and that it—or access to it—is compatible with the remainder of the NMCI computing and network infrastructure. An additional option would be to retire the application for the good of the service if it is deemed that the risk of using it outweighs its usefulness. This decision will most likely need to come from a higher authority, such as the Echelon II CIO or the DON CIO.

2.2.1.4 NCMI Lockdown Policy

NMCI lockdown policy disseminated through AD and enforced via GPOs is highly restrictive and requirements go beyond that of simple Windows 2000 certification. Essentially, the application may be subject to browser configuration changes via GPO that may affect how, when and if an web application can be used via an NMCI standard web browser. Typical GPO browsers configuration items can include but are not limited to browser user interface, connection, URLs, security (e.g.

security zonrs/privacy settings) and internet services default programs (e.g. default calendar, default html editor, etc).

2.2.1.5 Gold Disk Compatibility

An NMCI workstation application must maintain Gold Disk (baseline software applications included on Windows 2000 NMCI workstation) compatibility (see current Gold Disk definition at http://www.nmci-isf.com/downloads/Gold_disk_contents.pdf).

2.2.1.6 User Interface Specifications

User interfaces to applications must meet all current and DoD policies, processes, procedures, and standards (DII-COE, C4ISR-AF, JTA, DISCAP, Section 508, etc.).

2.2.1.7 NMCI Group Policy Objects

The Directory Services Team sets the desktop and application authentication standards. Developers need to contact this group when creating or modifying applications to obtain information on how to access the AD. Developers can call the NMCI Help Desk and request support from the Directory Services Team.

- AD domain administrators are able to set and modify group policy, but only under approval of direction of Navy and Marine Corps policies and NMCI group policies.
- Developers must modify applications to comply with group policy and lockdown.
- Developers must go through re-certification processes if their applications fail certification testing.
- Developers need to produce test plans that include the steps, data, and logical conditions necessary to trigger required authentication processes (Lightweight Directory Access Protocol (LDAP), AD, file sharing, file writes, etc.) to ensure group policy, lockdown, and security areas are thoroughly examined by the certification and Directory Services Teams.
- The Directory Services Team initiates a collaborated process to request group policy and/or lockdown changes (relax registry or file permission accesses) for applications that cannot be changed to meet current group policy rules or have a good case for a group policy change.
- As an application is tested, it may need to be modified by the developer to address conflicts between the group policy and the application. Alternatively, the DAA may agree to modifications to the group policy to accommodate the existing application. It is the responsibility of the ISF to make changes to the group policy as approved by the DAA.

2.2.1.8 Terminal Services

From a "terminal services" perspective, legacy application "NMCI Thin Client" architecture supports Windows 32-bit applications. The Citrix components (Nfuse, etc.) can interoperate with NMCI's portal. This makes it possible to launch PC-based applications from the portal, display across the intranet, and appear to run locally while running remotely.

From a developer's perspective, the best standard to follow is Microsoft's guidelines for how to design and construct applications to run best in a "multi-user environment" such as an environment with terminal servers.

Microsoft guidelines:

Optimizing Applications for Windows 2000 Terminal Services and Windows NT Server 4.0, Terminal Server Edition

http://www.microsoft.com/technet/treeview/default.asp?url=/technet/prodtechnol/win2kts/maintain/optimize/tsappdev.asp

3. CERTIFICATION AND DEPLOYMENT – PHASE III

Integration of an application in the NEP and NMCI requires application developers to complete several review and test processes. This section defines the processes that a web-enabled application must undergo for inclusion into the NEP as well as the current certification process for non-web-enabled NMCI applications.

The TFWeb Service Certification process, both classified and unclassified, is designed to ensure application services meet the security and functional standards of TFWeb and the Government prior to implementation within the production NEP. When an application has completed this process, it is certified for operating across the NEP environment on the IT-21 infrastructure. Applications not transitioning into the NEP will follow the existing NMCI certification process.

The TFWeb certification process begins when a service owner/developer registers the application/service on the Open Source Site (OpSS) https://tfw-opensource.spawar.navy.mil. The Application Migration Customer Support (AMCS) Team, upon receiving automatic notification by the OpSS, will work closely with the service owner and provide feedback throughout the process described in this section. Contacts associated with the service will be able to view the status of the submission/test/deployment online at the same site where registration occurs. The NMCI process for certification is currently undergoing significant change. The major steps in the process and basic considerations are described in this section. For updated instructions on specific areas, be sure to check with the ISF Tools Database at http://www.nmci-isf.com.

3.1 NEP INTEGRATION PROCESSES AND TFWEB

3.1.1 NEP Service Certification Process

The NEP service certification process is designed to ensure application services meet the security and functional standards of TFWeb prior to implementation in the NEP.

This process addresses UFSs developed by a Navy application developer to integrate a single application into the NEP. It does not provide guidance for the integration of other Navy portals (constituent portals). The Navy Marine Corps Portal (NMCP) Policy Guidance Memorandum number 1 dated February 28, 2003, defines the NMCP as the "single integrated enterprise portal structure that promotes a knowledge-centric environment, which will:

- Limit duplicative investments in portal technology
- Improve efficiency and effectiveness in portal usage though reduction in investments, promotion of common best practices and facilitation of authoritative data sources
- Promote DON-wide process engineering
- Support functional and organizational collaboration across the DoN

Refer to Figure 21: NEP Service Certification Process for a flowchart that follows this process.

While not required, NEP developer test portals are available that allow developers to develop and test services in environments similar to the NEP production systems prior to service certification process. The use of the development portals has been beneficial in quickly minimizing the integration issues that may be discovered. Developers can request access to these resources using the "Developer Test Site Registration" portlet on the Developer Resources Workplace in the OpSS.

A checklist summarizing the requirements for migration is in APPENDIX M: NEP Developer's Integration Checklist.

The NEP certification process begins when a service owner/developer contacts the designated Echelon II AMCS representative (see APPENDIX D: Points of Contact). The Application Migration Customer Support (AMCS) Team, comprised of TFWeb Navy personnel, will assist Navy organizations with the certification process. The service is reviewed by AMCS to verify that it has been validated by the appropriate FAM, does not overlap existing services, and supports the use of authoritative data sources. The AMCS and Application Migration Technical Support (AMTS) Teams provide assistance to service developers in making service changes and determining the documentation required for migration. Once the service is ready to migrate, the service owner submits a migration package online using the "Submission Package" portlet on the Developer's Resources Workplace in the OpSS, which AMCS reviews for completeness and approves.

After receiving the approved migration package from AMCS, the TFWeb Test Team performs testing within the IT-21 SIPRNET lab for classified services and the IT-21 NIPRNET environment for unclassified services. The TFWeb Test Team communicates any issues encountered during testing to the developer and AMCS so they can be resolved and testing restarted.

Once testing is completed, the application is certified and submitted to the NEP's IT Governance board for final approval prior to migration into the NEP.

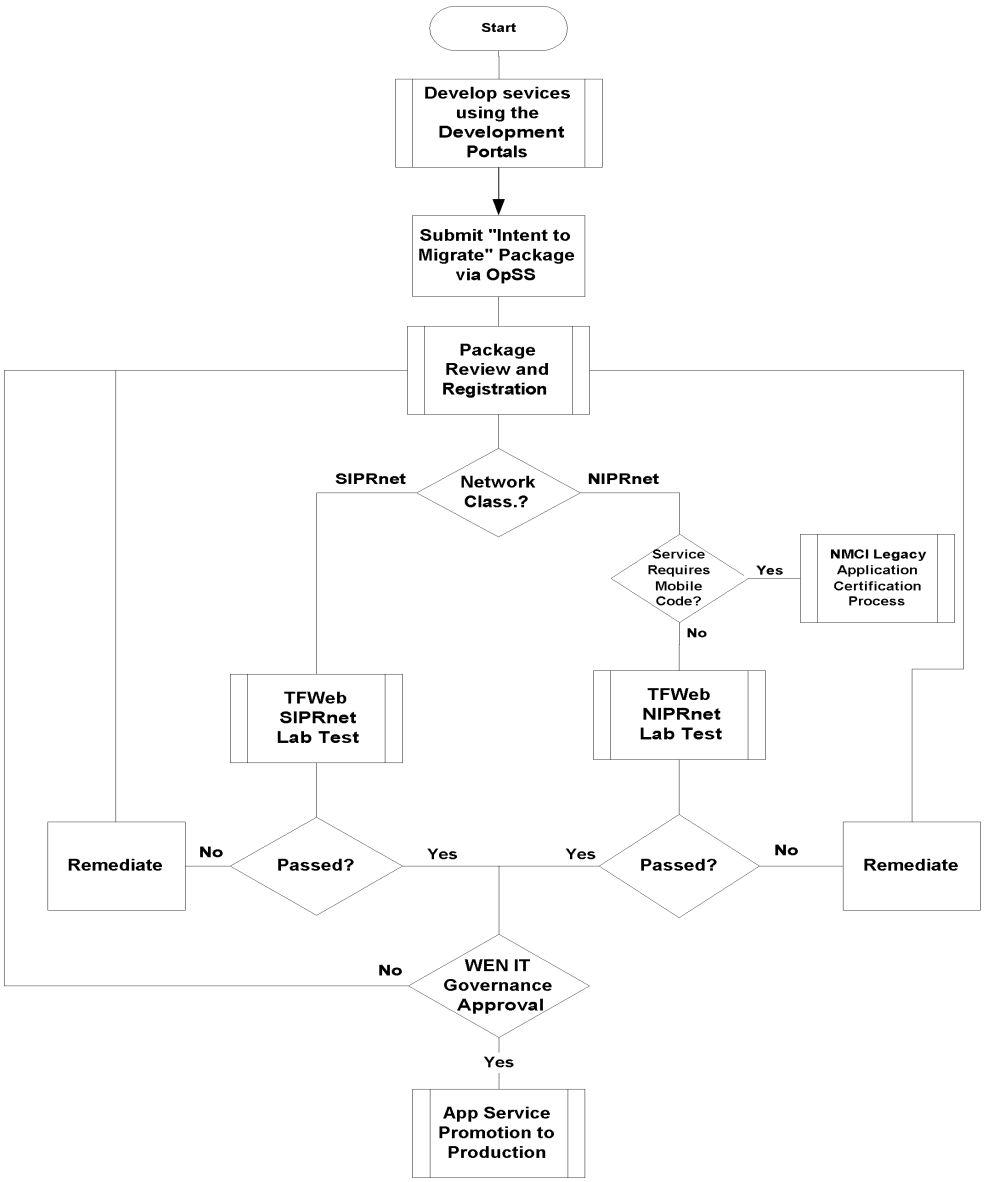


Figure 21: NEP Service Certification Process

3.1.2 Information Assurance Certification and Accreditation (IA C&A)

As directed in DoD Directive 8500.1 Information Assurance, para 4.13, "All DoD information systems shall be certified and accredited in accordance with DoD Instruction 5200.40." DoD Instruction 5200.40, DoD Information Technology Security Certification and Accreditation Process (DITSCAP), provides a roadmap to the certification and accreditation (C&A) process, and is further explained by DoD Instruction 8510.1-M, the DITSCAP Application Manual. As shown in the figure below, the DITSCAP process is designed to be adaptable to any information system, computing environment or mission, and is a living process. Each phase is thoroughly explained in DoD Instruction 5200.40 and DoD Manual 8510.1-M, DITSCAP Application Manual. All information relevant to C&A is collected during the DITSCAP process and documented in the System Security Authorization Agreement (SSAA). This document, which includes C&A evaluation results and a vulnerability assessment, is used by the system's Designated Approving Authority (DAA) to grant Interim Authority To Operate (IATO) or Authority to Operate (ATO).

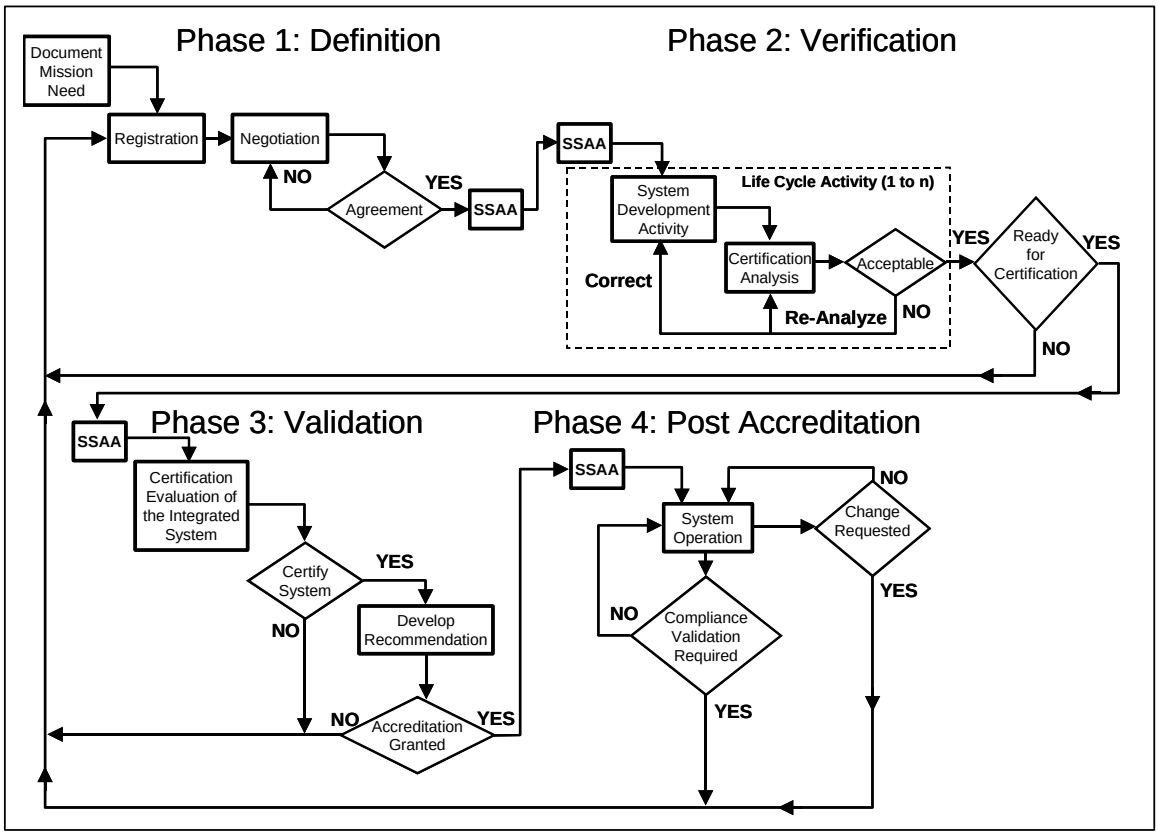


Figure 22: Overview of DITSCAP Phases *

***Reference:** DoD Manual 8510.1-M, Figure C2.F1., Page 28, http://www.dtic.mil/whs/directives/corres/html/85101m.htm

It is the responsibility of the service owner or developer to obtain an IATO or ATO through the DITSCAP process for each application prior to submitting a migration package for the TFWeb environment. Please see the Information Assurance Manager (IAM), formerly the Information System Security Manager (ISSM), representative for your command for more information on this process. Directives and instructions relevant to DITSCAP can be found on-line at: http://iase.disa.mil/policy.html, under the link "DITSCAP" or http://www.dtic.mil/whs/directives/ . Additional Navy-specific Certification and Accreditation guidance is contained in IA Pub 5239 Vol I, II and III, found on-line at: https://infosec.navy.mil/Documents/ under the tab "Navy."

The IATO/ATO must specifically address the service being migrated, and must be accompanied by appropriate DITSCAP documentation (the SSAA). In general, an IATO or ATO granted to an entire site or command is not acceptable unless the documentation demonstrates that the service migrated has been specifically tested and assessed. The IATO/ATO must be valid for at least six months after the date of submission. An application whose IATO/ATO expires will be removed from the NEP until a new IATO/ATO is received. Certification requirements that are required for the IATO/ATO are set by the DAA of the command sponsoring development or the command at which the software is hosted.

3.1.3 Service Migration Package Submittal

Services must be registered in the DON CIO Application and Database Management System (DADMS) at https://www.dadms.navy.mil. Input of database schema into the DADMS is encouraged. Once this information is received, the FAMs and AMCS work together to rationalize the service.

The migration package is submitted via online forms located on the OpSS by the application owner. The package is composed of three documents: the Migration Plan, Registry Metadata, and the Test Plan. Some of the information required includes application name, acronym, classification, taxonomy category, and a brief description explaining the functionality of the application. For assistance with the process, both a sample migration package and a listing of the appropriate AMCS POCs can be found at the TFWeb OpSS at https://tfw-opensource.spawar.navy.mil.

As applications are submitted, tested, and deployed to the NEP, annotating the application's version and providing a brief synopsis of changes from the previous release will be needed in order to provide seamless access to the application during version upgrades. This version-control requirement becomes even more critical as applications are converted into web services, where interface/algorithm changes could affect dependent applications/services.

3.1.4 NEP Taxonomy

NEP services are organized by operational and functional taxonomies. This allows users to easily find the services they desire based on common categories.

3.1.4.1 Functional Taxonomy

The functional taxonomy parallels the categories of functional data described in SECNAVINST 5000.36. Application owners should identify a single category to which their service primarily belongs. The appropriate category is determined by service function, not the mission of the service owner's organization. Therefore, most organizations will have applications in administration, manpower, and personnel in addition to their primary mission area. The categories are:

- Acquisition
- Finance
- Civilian Personnel
- Administration
- Manpower and Personnel
- Intelligence and Cryptology
- Logistics
- Readiness
- Command, Control, and Communications
- Information Warfare
- Allies

- Modeling and Simulation
- Weapons
- Training
- Resources, Requirements, and Assessments
- Scientific and Technical
- Test and Evaluation
- Medical
- Naval Reserve
- Meteorology, Oceanography, MC&G
- Religious Ministries
- Naval Nuclear Propulsion

3.1.4.2 Operational Taxonomy

The operational taxonomy is based on the organization of shipboard units and staffs. This provides an extra layer of granularity in areas such as administration, training, and shipboard operations. The operational taxonomy consists of eight primary areas or "N-codes". Each of these areas contains several sub-areas. The categories of the operational taxonomy are:

N1: Human Resources and Service Support

Military Personnel
Civilian Personnel
Administration
Medical
Dental
Chaplain
Legal
Inspector General
Public Affairs
Reserves

N2: Intelligence

Resources
Cryptology
Imagery

N3: Operations & Security

Common Operational Picture
Surface Operations
Submarine Operations
Air Operations
METOC

Information Operations
Security
Scheduling
Navigation
Information/Knowledge Management
Exercises

N4: Fleet Readiness & Logistics

Shore Maintenance
Afloat Maintenance
Aviation Maintenance
Configuration
Environment
Safety
Supply
Contacting
Facilities
Reporting

N5: Plans & Policy

N6: C4

COMSEC
RF Communications
Space
Networks
Configuration
Status
Training

N7: Training

Quota Control
E-Learning
Tracking
Lessons Learned

N8: Requirements & Assessments

Operational Testing
Certifications
INSURV
Audits
Inspections
Assessments

3.1.5 NEP Service Rationalization

Service rationalization is conducted by the FAMs and AMCS based on the information provided in DADMS, the migration package, and additional information as required from the service owner.

As part of the rationalization process, the service is compared to a PEO-IT managed Interim Standards list. The Interim Standards list was compiled by TFWeb, PEO-IT, NADTF, and the USMC to guide migration of services and applications to the NMCI and TFWeb environment. Services listed in the current Interim Standards list are automatically permitted to migrate to the TFWeb environment. All other services are checked for duplication of functionality, data sources, and user bases with services currently in the NEP and those in the migration process. If significant duplication is found, an analysis

is performed of the competing applications by AMCS and the results are submitted to the NEP IT Governance Board for resolution.

The ISF Tools Database is also used for rationalization. It includes all applications currently targeted for migration to NMCI and their latest status. It is expected that rationalization is rapidly completed after registration to provide developers rapid feedback on any required refocus or change to their service prior to making a significant investment in service migration.

3.1.6 NEP Service Migration Package

The service migration package should be submitted via the OpSS and should address the requirements listed in the following section. Parts of existing documents may be submitted to prevent unnecessary duplication of effort. A sample migration package can be found at the TFWeb OpSS https://tfw-opensource.spawar.navy.mil. Application and database owners creating web services for the NEP must ensure their applications/databases comply with all Navy and DoD directives for networked systems. Access through the NEP does not preclude or negate responsibility for compliance with Information Assurance, Section 508, and other all other applicable directives. The submission should be organized to provide the following information:

- Service Registry Metadata (see Table 18: Service Metadata Information)
- A temporary user login & password with access to non-administrator portions of the application to be used by the TFWeb Test team.
- Test Plan
- Migration Plan to Content Integration

If the service requires mobile code components, see APPENDIX N: Navy Mobile Code Policy, and 3.2 NMCI Integration Process and ISF for additional guidance. Contact AMCS for any additional migration requirements.

A checklist summarizing the requirements for migration is in APPENDIX M: NEP Developer's Integration Checklist. The test plan should cover the major functions of the service. The purpose of the test plan is to provide tests and expected responses to ensure that a tester without previous knowledge of the application can appropriately use an application. An incomplete test plan or missing login and password (if required) will lead to significant delays in certification of the service.

The migration plan should address the timeframe and intermediate steps planned for migration of the application to web services implementation. Currently, unmet concerns or requirements should be addressed in this document for TFWeb resolution. The developer and resource sponsor must regularly update the migration plan to reflect funding decisions in the Program Objective Memorandum (POM) process. POM submissions must reflect the commitments made in this document.

3.1.7 Service Registration Metadata

3.1.7.1 Service Registry Metadata

Service metadata is information stored in the service registry that describes the service itself and how the portal accesses it. A service owner can register multiple services per application. The information in Table 18: Service Metadata Information must be captured for each service registered.

Table 18: Service Metadata Information

Item	Description	Example	Enter Value Here
Service Name	A short, concise name that logically describes the service.	GetExamResults	
Service Description	A brief description of the functionality and/or information provided by the service.	A web service that retrieves enlisted exam results for the user.	
DADMS ID	The system ID in the DADMS that identifies the parent application for this service.	1234	
Operational Taxonomy	The operational taxonomy category under which the service is listed (See Section 3.1.4.2).	Human Resources/Personnel/Military	
Functional Taxonomy	The functional taxonomy category under which the service is listed (See Section3.1.4.1).	Administration	
Version	The version of the service that is deployed on the host application\web server.	Version 1.0	
Target Users	Allows a developer to select the user profiles to see the service from their default view. Select from any of the following: - Active Duty Personnel - Retired Personnel - Government Civilians - Reservists - Contractors - Dependents - Joint/Coalition Forces	Active Duty, Reservists	
Target Portals	A description of the group of portals to which this service should be replicated.	GW battle group	
Binding Type	Describes the type of binding the portal uses to communicate with the service. Select from HTTP or WSDL.	HTTP	
HTTP Access Point URL	If the binding type is HTTP, the service owner must submit the fully qualified URL to the entry point of the service. Submit one URL for each	http://bupers.navy.mil/Exams/Results.asp (sample link)	

Item	Description	Example	Enter Value Here
	instance of the service.		
WSDL URL	If the binding type is WSDL, the service owner must submit the fully qualified URL to the WSDL that describes the web service. Submit one URL for each instance of the service.	http://bupers.navy.mil/Exams/Results.wsdl (sample link)	
Send PRI	Flag indicating whether or not the PRI data is added to the request sent to the service. The method used to pass the PRI data is dependent upon the type of invocation used. The PRI data message is attached to the HTTP header for standard HTTP calls and if SOAP is used is embedded in a SOAP header. This option is applicable only with rewritten URLs.	Yes (default is No)	
Send Identity	Flag indicating whether the portal passes the HTTP authentication header from the original client request to the service being called. This option is applicable only with rewritten URLs.	Yes (default is Yes)	
Insert Style	Flag indicating whether the portal inserts the appropriate portal CSS into the HTML return stream, thus automatically handling the portal look and feel. This option is applicable only with rewritten URLs.	No (default is No)	
Rewrite URLs	Flag indicating whether the portal attempts to rewrite URL references in the UFS response stream. This option is one of the techniques available in order to enable options for SendPRI, SendIdentity, InsertStyle, or ContentMIMEType. See APPENDIX H: URL Rewrite Guidelines for more information.	Yes (default is Yes)	
Content	Optional. The Multipurpose	Text/XML (others include	

Item	Description	Example	Enter Value Here
MIME Type	Internet Mail Extensions (MIME) type of the content being returned to the portal from the service. Specifying the MIME type on this parameter can increase efficiency as the logic that attempts to determine the MIME type can be avoided. This option is applicable only with rewritten URLs.	text/HTML)	
Render XML	Optional. Flag indicating whether the portal attempts to render XML using an XSLT stylesheet reference embedded in the XML document. Setting this to N allows a service to pass the raw XML to the client to support client-side rendering. This option is applicable only with rewritten URLs.	No (default is Yes)	
Generate Cookies	Optional. Controls whether the portal will generate a session cookie with the information necessary to allow re-invocation. This is intended for use when a forward proxy is used instead of the URL rewrite proxy. This option is not currently implemented.	Yes (default is No)	
Required Parameters	A list of parameters the service requires to process. The parameters will be stored with the service metadata in the registry.	Location=Norfolk	

3.1.7.2 Service Contact Information

Contact information is collected at the service level and stored in the service registry. A service owner should register service contact(s) for each registered service. Contacts identified as “Customer Service” should be able to modify the service access controls and provide basic assistance in service access. Contacts identified as “Technical” should be able to provide detailed information on the service and its interfaces. All contacts associated with the service will be able to upload the online forms and view the submission status through the OpSS.

Table 19: Service Contact Information Description

Item	Description	Example	Enter Value Here
Name	Name of the contact person	Joe Developer	
Contact Type	Relationship contact has with service. Valid entries are Manager, Customer Service, and Technical.	Customer Service	
Phone	Phone number of the contact	(555) 555-5555	
E-mail	E-mail address of the contact	Joseph.developer@navy.mil	
Description	Role or title of the contact	Senior Developer	
Address	Mailing address of the contact	123 Anystreet, Anytown VA	

3.1.8 Service Lifecycle Management

After services have been migrated to the NEP, it is expected that they will continue to be updated and modified in response to user requirements. Service changes requiring modification to the NEP must be submitted to the test team via AMCS for additional testing. This submission should include a revision history with a brief synopsis of changes and will be appended to the original migration package. Service changes that require retesting include the following:

- Modification to the URL of a site
- Installation of a new PKI certificate
- Change to the service architecture
- Modification of service security

Minor updates to content or business logic do not require retesting. Retesting, if performed, will be required only on the modified portion of the service.

3.1.9 TFWeb Test Processes

TFWeb lab testing ensures that application services function appropriately within the portal environment and that they adhere to TFWeb standards. There are currently two TFWeb test labs, a SIPRNET lab and a NIPRNET lab.

3.1.9.1 Lab Testing Process

Packages are submitted for certification testing using the following process:

The AMCS reviews and accepts migration packages and then forwards them to the lab for certification testing. Inadequate migration packages will not advance to the testing lab.

The service will be scheduled for certification testing.

The TFWeb Test Team registers the service in the Quality Assurance System (QAS) service registry, which creates the global unique identifier (GUID) service key that is used by the portal to reference the service.

The TFWeb Test Team performs the tests, utilizing the provided test plan with test cases.

If the application fails any test case or if its performance impacts the NEP environment, the application will not be certified. In this case, the service owner and AMCS will be notified with specific reasons for failure.

(NIPRNET Only) If a service requires modification of the desktop configuration (i.e., plug-ins, active-X controls, etc.), then NMCI requires that desktop applications go through an additional process in order to certify the security of the service. The NMCI Application Certification Process is outlined in 3.2 NMCI Integration Process.

When the service passes testing in the IT-21 environment, a notification letter will be sent to the service owner and AMCS.

Once testing is complete, the service is certified and submitted to the NEP IT Governance board for final approval prior to deployment into the NEP.

3.1.10 NEP IT Governance

Services that fail suitability criteria and/or portions of the testing may be functionally displaced by another service by the time they are ready for migration to the production portal. Testing may also demonstrate substantial overlap with another service or organizational issues that prevent immediate migration of the service. Final approval of migration is currently a function of the TFWeb ESG. This approval may be delegated to a lower level based on service compliance with TFWeb standards.

3.1.11 NEP Service Deployment

After a service has completed the testing and certification process, it is ready to deploy into the production NEP systems. This process is referred to as "service promotion to production" in Figure 21: NEP Service Certification Process. This process requires careful coordination between the NEP engineering team and the service owner/developer to ensure a smooth transition. The goal is to provide each NEP system with a complete set of services and to avoid any broken links.

In some cases, the service testing may have been done against a developmental service instance and it would not be appropriate to deploy those service instance URLs to production NEP systems.

For ashore-based service instances, you should become familiar with the NEP installation plan and compare it to the installation schedule of your production system(s). Newly certified services are replicated as they become available.

Afloat NEP installations are timed according to ships availability schedules. After the initial installation, newly certified services are replicated as they become available.

For both afloat and shore services, provide production service instance deployment URLs (and other production metadata) to the NEP engineering team as soon as they are known.

The service metadata field *Target Portals* is particularly important for accurate service deployment to the correct production portals. If it is known that specific ships should receive the service, then a list of hull numbers should be provided.

The Functional and Operational taxonomies determine how users will locate a specific service in the NEP. The service name must be descriptive enough that users can distinguish the service from other services that appear under the same taxonomic categories.

There is additional planning to consider if a service is integrated with SSO. Specifically, additional configuration must be done in each production NEP system.

3.2 NMCI INTEGRATION PROCESS AND ISF

Integration of an application into the NMCI requires the completion of several processes, most of which involve interaction with the ISF. The specific processes for interaction with the ISF are under development, and are documented in Supplement A, the NRDDG. The NRDDG has a "Design and Development" section that was comprised collaboratively between the NMCI Program Management Office PMO and the NMCI contractor, the ISF. The design and development section contains standards and programming practice reference that can assist a developer creating and submitting to the ISF an applicationt that complies with the NMCI specific rules and policy.

3.2.1.1 Certification and Accreditation

The DoD Instruction (5200.48) 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 standardizes the C&A process for single IT entities that lead 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, and III). A main tenet of DITSCAP is tailorability. The level of effort to accomplish C&A can be customized to the application seeking accreditation based on customization, application/system complexity, mission criticality, the mode of operations of the environment that the application is functioning in, etc. Detailed information can be found in the NMCI Connection Approval Process (NCAP) available at http://www.nmci-isf.com/NCAP.doc or via https://infosec.navy.mil. Additional information can be found in DoD – DITSCAP and Navy IA Pub 5239-13 (Vols I, II, and III).

3.2.1.2 Authoritative Data Source

In accordance with SECNAVINST 5000.36, Authoritative Data Sources (ADSs) shall be designated by the appropriate functional data manager (FDM). Information about ADSs shall be maintained in the DON Application and Database Management System (DADMS) at https://www.dadms.navy.mil.

3.2.1.3 NMCI Application Hosting

The Legacy Systems Support CLIN 0029 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.

For further information, refer to http://www.nmci-isf.com/clinlist.pdf for available hosting options.

3.2.1.4 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.

3.2.1.5 Approvals

Approvals to operate applications on NMCI are granted by ISF's certification process defined at http://www.nmci-isf.com. Certifications are issued by submitting media for testing, and a certificate and ATO are issued to the application owner once all the requirements have been met. For more information, refer to http://www.nmci-isf.com.

3.2.1.6 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 the following:

- 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.nmci-isf.com/catalog.htm

The S&T NMCI seats are the only seats designed for development activities but must not be connected to the NMCI network during development to prevent adverse impact to the NMCI environment.

3.2.1.7 Accreditation Plan

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

3.2.2 Before Visiting NMCI for an Engineering Review

The process of transitioning applications to NMCI entails a set of interrelated processes that impact various components and the ISF. This guide seeks to communicate the transition requirements and expectations with the objective of enabling the customer to effectively plan and efficiently execute his or her transition to NMCI.

3.2.2.1 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 this development guide

3.2.2.2 Security Certification and Accreditation Process

NMCI is required to abide by the DITSCAP process. As such, NMCI will be "accredited" per the DITSCAP. As described in VI.B.3 Security C&A, the C&A efforts integrated into the application should be appropriately documented in the Key Elements of the SSAA for review. These Key Elements are as follows:

- Definition and Appointment of IA personnel (DAA, 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

3.2.2.3 NMCI Security Certification and Accreditation Process (NSCAP)

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

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

3.2.2.4 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 MS Office applications continue to operate?); any consideration for prototype/pilot testing; the steps, data, and logical conditions necessary to trigger programmed authentication processes (LDAP, AD, file sharing, file writes, etc.) to ensure Group Policy, Lockdown, and Security areas are thoroughly examined by the Certification and Directory Services teams. Developers must ensure logon IDs have the same access rights as end-users, not developers.

The Microsoft Developer Network (MSDN) web site provides various testing and certification documents and tools that can be used to test desktop and server applications, but the final authority on certification testing is the responsibility of the ISF and will be handled by the ISF certification lab (see http://msdn.microsoft.com/certification and http://www.nmci-isf.com/transition.htm for application testing references and checklists).

3.2.3 Certification Lab Activity

For familiarization and preparation of the Application Certification Process, developers can initiate several processes and documents. The public NMCI web site at http://www.nmci-isf.com/transition.htm has links to online documents for the following:

- Legacy Application Transition Guide (available at http://www.nmci-isf.com/transition.htm#Transition)
- 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 in the following section.

3.2.3.1 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 where experienced users are not available for questions.

3.2.4 Certification Lab Process

This section describes the classified and unclassified lab certification processes, including PoP-in-a-Box (PIAB), a mobile server that approximates the NMCI environment and 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.

The steps for Application Integration and Testing (AIT) processes are illustrated in Figure .

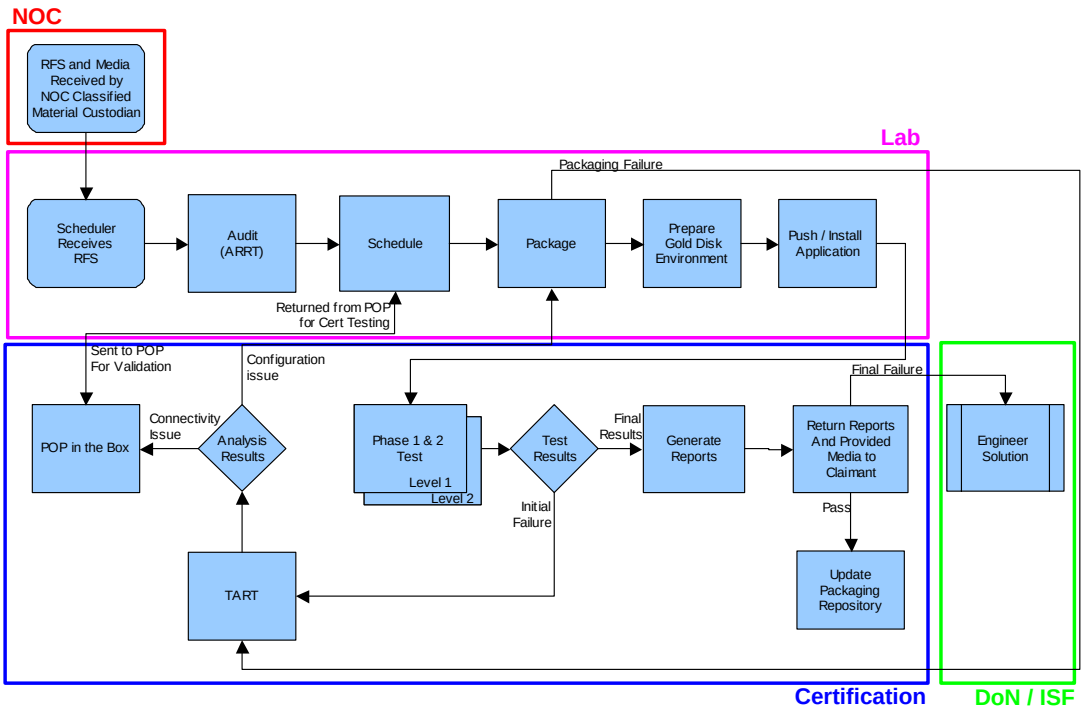


Figure 23: Application Integration & Testing Lab (AIT) Process

This process is applied 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.

3.2.4.1 Application Integration and Testing Process Steps

Customer/ISF Initiation - Initiation of the certification process:

- 'As Is' environment – prior to cutover to NMCI, the ISF, PMO, and customer sites work together to identify and collect data on legacy applications, rationalize lists, and then submit an RFS for each application.

- 'To Be' environment – after cutover to NMCI, the acquirer of the application may introduce new or emerging applications by submitting an RFS via the proper chain of command and issuing a CLIN 0029 task order for certification testing.

3.2.4.2 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 by the appropriate media, key/license, and any customer test scripts or special instructions, if applicable.

3.2.4.3 Audit (ARRT)

A review process to ensure that all informational and material requirements have been met for certification processing. This process is conducted internally at the certification lab by the Application Rationalization and Review Team (ARRT).

3.2.4.4 Scheduling

After a successful audit, the RFS is scheduled to a resource/cell. If there is a need to prioritize an RFS, this should be done by contacting the PMO, who then conveys the priority need to the ISF/Application Integration & Testing AIT lab.

3.2.4.5 Validation

Validation will be conducted on site utilizing the PIAB engineering tool. Some applications require site connectivity in order to validate application functionality and/or connectivity/security compliance. The AIT lab may send an application to PIAB for pre- or post-testing in the lab.

3.2.4.6 Certification Pass/Fail

The responsible certification team manager generates an NMCI technical certification letter, NMCI application release notes, and NMCI certification certificate stating the results of the certification process.

3.2.4.7 Parties to the Process

Following are parties to the AIT 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 Design Activity (CDA) – The Government application developer.
- (Classified Applications only) NOC Classified Material Custodian – The ISF (Raytheon) individual responsible for receipt and accountability of classified material at the NOC.
- ARRT – The team responsible for providing an initial review/audit of the RFS and ensuring all informational requirements have been fulfilled.
- Lab Scheduler – 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.
- TART – 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 – AIT personnel responsible for assisting, monitoring, and coordinating the application-gathering effort.

- PoP-in-a-Box (PIAB) – 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.

3.2.4.8 Developer Responsibilities

Developers are responsible for performing the following:

Required to follow the certification processes and forms to have their applications authorized to be operating within NMCI.

Must follow these processes and related life-cycle processes any time 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 site) at the certification lab during certification steps but are encouraged to do so if they desire.

For PIAB testing, developers are responsible or involved in the pre-/post-certification processes and documents. Developers are responsible for providing application test scripts, application installation instructions, user IDs, and license keys, and for being present at installation (if necessary), etc.

3.2.5 Before Deployment/Migration

3.2.5.1 Help Desk Procedures

Per the Naval message released by CNO N09T, date time group 252250Z FEB 02, "THE CDA HAS PRINCIPAL RESPONSIBILITY FOR DESIGN, DEVELOPMENT, DOCUMENTATION, AND LIFE CYCLE MAINTENANCE OF APPLICATIONS, INCLUDING INITIAL PRODUCT DELIVERY AND DISTRIBUTION OF UPDATES. ADDITIONALLY, CDA(S) RESOURCE AND MAINTAIN HELP DESK SERVICES FOR THEIR APPLICATIONS." Developers must ensure that the NMCI Help Desk is properly notified and prepared to handle user inquiries on their applications and the help desk should escalate software-related tickets as appropriate. See APPENDIX D: Points of Contact for the NMCI Help Desk contact information. As of this writing, no official help desk specific information is being collected, but a knowledge base is being developed. It is recommended that application owners ensure the NMCI Help Desk can properly escalate application-related calls to CDA help desks as needed.

All Navy claimants and Marine Corps organizations are encouraged to establish an application help desk and coordinate efforts through the NMCI PMO and STEM to assist their activities and sites with these transition efforts.

It is recommended that application developers submit the basic information to the NMCI Help Desk, including but not limited to the following fields:

General Questions:

- What are the program files for the application?
- What file extensions do the data files for the applications have?
- List the top five (5) issues that the application has and the fixes for those issues:
- Are there specific settings for the application depending on the physical location of the user?
- Do users need to have full control of any of the files or folders associated with the application?
- Is there a program that runs in association with the application?
- Printer Setup:
- What type of printer port does the application use?
- Network specifications:
- Is there a specific drive specification for the application?
- What steps does a user need to do to be able to connect to this application in the NMCI environment?
- Required Updates:
- Does this application require any updates? (monthly, bi-monthly, yearly)
- If application requires updates, how are we to obtain them and who is the source for the updates?
- RRAS Settings:
- What changes need to be made, if any, when a user is using NMCI-ISF RRAS for connection to the intranet?

Table 20: NMCI Help Desk Recommended Developer Fields

Category	Field
Description	
Contact Information	Help Desk Phone Number Misc Contact Information Application Website Site for Patches Site for Drivers Email Notes Misc Sites
Application	Name Radia Instance Name Commands using Applications Known Problems Web Drive Mapping Information Used with Other Applications?
Installation	Who can install it? Other Notes Permission Required
Configuration	Base Specific Information? Other Notes Separate Print Driver
Instructions	

- Note:

-

- Table 20: NMCI Help Desk Recommended Developer Fields enumerates common fields in enterprise help desk support software. The (*) fields are typically required. The NMCI Help Desk will field calls on an application to determine if the problem is infrastructure related, correct any infrastructure issues, log the incident, escalate unresolved tickets to the application owner or non-NMCI support, and then close the trouble ticket. Because advanced or usability questions about an application will be escalated to the POC associated with the application, it is the responsibility of the application owner or maintainer to notify the help desk of any changes that would impact NMCI Help Desk initial support of an application.

- When working with the EPMO, in addition to the items in paragraph 3.2.5.1 above, the NMCI Help Desk will need to know the following:

- Expectation of the level and the extent of troubleshooting to be done

- Escalation path outside of the NMCI Help Desk

- Applicable SLAs

- Availability and necessaty of Agent training

- Extent/Scope of deployment

- Rollout Plan

3.2.5.2 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 is available at no additional cost. Service level agreement (SLA) 17 defines training requirements.

3.2.5.3 Backup and Recovery

Developers must create and test an appropriate back-up and recovery (B/R) process and identify an up-to-date B/R plan for their systems. The ISF will provide back-up services for all network-stored data. Back-up of individual workstation hard drives is the responsibility of the end user.

3.2.6 Deployment/Migration

3.2.6.1 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/Marine Corps are based on the NT4 Terminal Server operating system. The existing Navy/Marine Corps architectures and assumptions are likely incompatible with the “NMCI Thin Client Architecture”. For example, existing Navy/Marine Corps thin client architectures include security, permissions, and domain 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 its own "thin client architecture," so leveraging solutions across sites/commands/bases may not be possible.

3.2.6.1.1 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:
If the claimant runs applications on terminal services today and the claimant wants to perform his or her own server support, then 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.

Legacy Server Support:
If a claimant runs applications on terminal services today and the claimant wants the ISF to support pre-existing servers, a task order under CLIN 0029 must be executed for re-engineering and hosting services.

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.

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.

3.2.6.1.1.1 Execution Results CLIN 0029 for Applications on Terminal Services

- Determine compatibility with Windows 2000 Professional and Windows 2000 Terminal Services.
- Determine how many sessions a terminal server can support.
- Determine reusability of existing hardware and software.
- Determine network connectivity and security requirements.
- Determine ID, group, and OU requirements.
- Determine if portal integration is necessary.
- Determine performance measurements.
- Determine ongoing costs, if any.

4. REFRESH AND RETIREMENT – PHASE IV

This section provides key considerations to roll out application updates and initiate application retirement. This section will be expanded as the FAM application rationalization efforts expand to eventually cover application retirement.

4.1 SYSTEM CHANGES

Following are procedures for system changes.

4.1.1 Emergency Production Fixes

Emergency production fixes may be authorized only if the problem is critical, may jeopardize safety, or 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 is 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 for introducing an emergency fix include the following:

- Investigate the problem and determine the cause, including both system and process-related issues.
- Develop the fix.
- Test fix to ensure it meets the requirements.
- Regression test to ensure all related functionality has not been impacted by the fix.
- Enter fix into the configuration management process where it can be tracked and followed through the formal release process.
- Include fix in the next formal release.

4.1.2 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.

4.2 SYSTEM RETIREMENT/SUNSET

Detailed processes and procedures for shutting down an application currently do not exist. However, the FAM application rationalization processes seek to reach keep-or-retire decisions that will lead to application retirement activities. Key to successful retirement is meticulous planning to allow the users to transition to other applications that will support their business requirements. Retirement includes a comprehensive communication plan including ISF, help desk, users, 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.
- Deploy replacement application, fully supported by training and communication plan.
- Transition users to replacement application.
- Stop the application from running on the retirement date.
- Make a backup copy or an archive to store for historical purposes.
- Remove the application and uninstall any additional software required to run from all applicable machines.

Coordinate with the NMCI help desk to remove the application from support lists and close any outstanding tickets.

Coordinate with DADMS to remove application from the FAM portfolio.

APPENDIX A: Glossary of Terms And Acronyms

Term	Description
3DES	Triple Data Encryption Standard
A&E	Architecture and Engineering
AACT	Advanced Application Certification Testing
ADSI	Active Directory Service Interface
AIT	Application Integration and Testing - Team that provides the environment for development and testing within the NMCI structure.
AMCS	Application Migration Customer Support (TFWeb term)
AMTS	Application Migration Technical Support Team (TFWeb term)
ATO	Authority to Operate
BAN	Base Area Network
BLII	Base Level Information Infrastructure
C&A	Certification and Accreditation
CA	Certificate Authority
CAC	Common Access Card (smart card)
CAST	Center for Applied Science Technology
CBT	Computer Based Training
CCB	Change Control Board
CDA	Central Design Activity
CDW	Certification Data Warehouse
CGI	Common Gateway Interface
CIO	Chief Information Officer
CLIN	Contract Line Item Number
CM	Configuration Management

Term	Description
CNO	Chief of Naval Operations
COI	Community of Interest
COM	Component Object Model
COM+	Common Object Model Plus
Common Identity	The user identification that is unique across the enterprise.
Portal Services	The logical set of common portal functions and services exposed and available to the User Facing Service Developer.
Content	The text, graphics, audio, video, services, and applications available at a web site.
COTS	Commercial-off-the-Shelf (software)
CRL	Certificate Revocation List
CSS	Cascading Style Sheets
CTR	Contract Technical Representative
DAA	Designated Approval Authority
DADMS	DON Application Database Management System (https://www.dadms.navy.mil)
Data Oriented Service	A software component that receives a request and optionally returns an XML Data Response. A Data Oriented Service may interact with portal services and other services published in the service registry.
DII	Defense Information Infrastructure
DISA	Defense Information Systems Agency
DITSCAP	Defense Information Technology Security Certification and Accreditation Process
DMIR	Data Management Interoperability Repository (https://www.dmir.navy.mil) Replaced by DADMS (https://www.dadms.navy.mil)
DMZ	Demilitarized Zone
DNS	Domain Name Server
DoD	Department of Defense
DOM	Document Object Model

Term	Description
DON	Department of the Navy
DOS	Data Oriented Service
DS	Directory Services
DTD	Document Type Definition
EAGLE	Enterprise Applications Group for Legacy and Emerging
EDS	Electronic Data Systems
EJB	Enterprise Java Beans - 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 and enable the developers to focus on actual business architect.
ESG	Executive Steering Group
FAM	Functional Area Manager
FAQ	Frequently Asked Question
FY	Fiscal Year
GOTS	Government-off-the-Shelf (software)
GPO	Group Policy Object
GUID	Globally Unique Identifier, also commonly known as a UUID
HI	Horizontal Integration
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IA	Information Assurance (Security)
IATO	Interim Authority to Operate
IATT	Information Assurance Tiger Team
ID	Identification
IE	Internet Explorer

Term	Description
ILS	Integrated Logistics Support
IM	Information Management
INFOSEC	Information Security
IOC	Initial Operational Capability
ISDN	Integrated Switched Digital Network
ISF	Information Strike Force
ISSM	Information Systems Security Manager
ISSO	Information Systems Security Officer
IT	Information Technology
IT-21	Information Technology for the 21st Century
ITC	Information Transport Cloud
J2EE	Java 2 Enterprise Edition - Introduced in 1995 by Sun Microsystems. It is an object-oriented language designed for the World Wide Web, similar to c/c++, in which the source is compiled into 'bytecode,' which is then interpreted by run-time environment.
Java	A general purpose, high-level, object-oriented, cross-platform programming language developed by Sun Microsystems [not an acronym].
JDBC	Java Database Connectivity
JITC	Joint Interoperability Test Command
JSP	Java Server Pages
JTA	Joint Technical Architecture
LATG	Legacy Applications Transition Guide
LDAP	Lightweight Directory Access Protocol
LOE	Level of Effort
MCSC	Marine Corps Systems Command
MCTN	Marine Corps Tactical Network
Metadata	Metadata describes how, when, and by whom a particular set of data was collected and how the data is formatted. Metadata is essential for understanding information stored in data

Term	Description
	warehouses.
MIME	Multipurpose Internet Mail Extensions
MS	Microsoft
NADTF	Navy Application Database Task Force
NAVSEA	Naval Sea Systems Command
NEP	Navy Enterprise Portal - The logical set of functional components that comprise the central portal infrastructure, including the portal, the service registry, and the portal services. The gateway to the Navy Enterprise Portal is https://portal.tfw.navy.mil/.
NEXT	Near End Cross-Talk
NIPRNET	Non-Secure Internet Protocol Router Network
NMCI	Navy Marine Corps Intranet
NOC	Network Operations Center
NRDDG	NMCI Release Development & Deployment Guide
NTIRA	Navy Tool for Interoperability and Risk Assessment
OAG	Operational Advisory Group
OCONUS	Outside Continental United States
ODBC	Open Database Connectivity
OpSS	Open Source Site, the TFWeb developer web site at https://tfw-opensource.spawar.navy.mil
OU	Operational Unit
PDA	Personal Digital Assistant
PEO-IT	Program Executive Office for Information Technology
PKI	Public Key Infrastructure
PM	Program Manager
PMO	Program Management Office
POC	Point of Contact

Term	Description
PoP-in-a-Box (PIAB)	Point of Presence In a Box – software testing tool that emulates NMCI network environment
Portal	The functional component of the Navy Enterprise Portal that is responsible for aggregating portlets.
Portal Client	A software application or hardware device that communicates with the Navy Enterprise Portal using the Portal Client Interface. Includes the set of web browsers, PDAs, and mobile devices.
Portal Client Interface	An HTTP(s) Request/Response initiated by a portal client to the Navy Enterprise Portal.
Portal Service Request	A request sent to a common portal service from a UFS.
Portal Service Response	A response sent from a common portal service to a UFS.
Portlet	The visible, active windows that end users see within their enterprise portal interface.
PPL	Preferred Products List
PRI	Portal Request Interface
QAS	Quality Assurance System (TFWeb test system)
QPL	Qualified Parts List
REST	Representational State Transfer, an alternate architectural style for developing web services
RFS	Request for Service (DITCO)
SDK	Software Developer Kit
Service Registry	The functional component of the Navy Enterprise Portal that stores metadata on UFSs and DOSs.
SGML	Standard Graphical Markup Language
SHC	Stakeholders' Council
SIPRNET	Secure Internet Protocol Router Network
SLA	Service Level Agreement
SME	Subject Matter Expert
SOAP	Simple Open Access Protocol

Term	Description
SOC	Security Operations Center
SOE	Standard Office Environment
SPAWAR	Space and Naval Warfare
SQL	Structured Query Language
SSAA	System Security Authorization Agreement
SSC	SPAWAR Systems Center
SSIL	System/Subsystem Interface List
SSL	Secure Sockets Layer
STEM	Site Transition Execution Manager
TARF	Technical Assistance Request Form
TART	Technical Applications Review Team
TCP	Transmission Control Protocol
TFW	TFWeb, Task Force Web
TO	Task Order
UAT	User Acceptance Testing
UDDI	Universal Description Discovery and Integration
URI	Uniform Resource Identifier
URL	Uniform Record Locator
User Facing Service (UFS)	A software component that receives a UFS request from the portal and returns a UFS response that formats the content for display (usually in a markup language such as HTML or WML) to produce visual output in a portlet. A UFS may interact with portal services and other services published in the service registry
User Facing Service Request	A request sent to a UFS from the Navy Enterprise Portal. There are currently two types of UFS requests: HTTP request and HTTP SOAP request.
User Facing Service Response	A response sent to the Navy Enterprise Portal from a UFS.

Term	Description
USMC	United States Marine Corps
UUID	Universally Unique Identifier, also commonly known as a GUID
VPN	Virtual Private Network
W3C	World Wide Web Consortium
WAN	Wide Area Network
Web Service	A software component that is described via WSDL, to which a reference can be published and located in a UDDI registry, and invoked via SOAP over HTTP(s).
WEN	Web Enabled Navy
WML	Wireless Markup Language
WSDL	Web Services Definition Language
WSEE	Web Service Execution Engine
WSRP	Web Services for Remote Portal
WWW	World Wide Web
XML	Extensible Markup Language. An extension/subset of Standard Graphical Markup Language (SGML) specifically designed for WWW dissemination and display of data. It is an open framework.
XSL	eXtensible Stylesheet Language
XSLT	eXtensible Stylesheet Language Transformations

APPENDIX B: References

Useful Hyperlinks

Table 21: Useful Hyperlinks (all external links)

Name	URL
NMCI Websites	http://www.nmci-isf.com http://www.nmci.navy.mil
General DoD Policies for Web Content	http://www.defenselink.mil/webmasters
DoD Mobile Code Policy	http://www.c3i.osd.mil/org/cio/doc/mobile-code11-7-00.html
DON Policy on the use of Extensible Markup Language (XML)	http://quickplace.hq.navy.mil/navyxml
DoD PKI Policy	http://www.c3i.osd.mil/org/cio/doc/may172001.pdf
Cookie/Privacy Policy	http://www.c3i.osd.mil/org/cio/doc/cookies.html
INFOSEC Web Site	http://infosec.navy.mil
Public NMCI Web Site	http://www.nmci-isf.com
J2EE	http://java.sun.com/j2ee/
.NET	http://microsoft.com/net/
UDDI 2.0	http://www.uddi.org/ http://www.oasis-open.org/committees/uddi-spec/
Web Services	http://webservices.org/ http://webservices.xml.com
WSDL	http://www.w3.org/tr/wsdl
Section 508 Compliance	http://www.section508.gov

Transition Teams

Table 22: NMCI Transition Teams

Name	Acronym
Site Transition Execution Manager	STEM
Enterprise Applications Group for Legacy and Emerging	EAGLE
Information Assurance Tiger Team	IATT
Navy Application Database Task Force	NADTF
Technical Applications Review Team	TART

Navy Messages

Table 23: Related Navy Messages

Originator	Message Date Time Group (DTG)	Subject
CNO WASHINGTON DC//N09T/N1/N2/N3/N4/N6/N7/N8/N093/N095/ N096//	R 252250Z FEB 02	NMCI LEGACY APPLICATIONS TRANSITION PROCESS//
PEO IT WASHINGTON DC//	R 261800Z FEB 02	ENTERPRISE LEGACY APPLICATION MANAGEMENT//
CNO WASHINGTON DC//N09T/N09W//	R 171442Z APR 01	NAVY ENTERPRISE PORTAL//
CINCPACFLT PEARL HARBOR HI//	R 050243Z OCT 01	NIPRNET PRIVATE WEB SERVER POLICY//
DIR NMCI & PMO//	R 242225Z MAY 02	NMCI PROCESS SUMMIT AGREEMENTS//

Important Standards

Table 24: Important Standards (all external links)

Standard - Version	Description	Standards Body	URL
HTTP 1.1	HyperText Transfer Protocol	IETF, June 1999	http://www.ietf.org/rfc/rfc2616.txt
URI	Uniform Resource Identifiers (URI)	IETF, August 1998	http://www.ietf.org/rfc/rfc2396.txt
HTML - 4.01	Hypertext Markup Language	W3C, Dec 1999	http://www.w3.org/TR/html401/
CSS - 1	Cascading Style Sheets (CSS1)	W3C, Jan 1999	http://www.w3.org/Style/CSS/
XML - 1.0	eXtensible Markup Language (XML)	W3C, October 2000	http://www.w3.org/XML/
Namespaces in XML	eXtensible Markup Language Namespaces	W3C, Jan 1999	http://www.w3.org/
XSL - 1.0	eXtensible Stylesheet Language	W3C, Oct 2000	http://www.w3.org/Style/XSL/
XSLT – 1.0	XSL Transformations	W3C, Nov 1999	http://www.w3.org/
XHTML - 1.0	eXtensible Hypertext Markup Language	W3C	http://www.w3.org/MarkUp/
ECMASCRIPT	JavaScript Standard from European Computer Manufacturers Association	ECMA, Dec 1999	http://www2.hursley.ibm.com/tc39/ecma262-3.pdf

Standard - Version	Description	Standards Body	URL
DOM – Level 2	Document Object Model	W3C	http://www.w3.org/DOM/
XSD – 1.0	XML Schema Description Language	W3C	http://www.w3.org/XML
SAML 1.0	Security Assertion Markup Language	OASIS – Nov 2002	http://www.oasis-open.org/committees/security/
SOAP 1.1	Simple Object Access Protocol	W3C, May 2000	http://www.w3.org/
UDDI 2.0	Universal Description, Discovery and Integration	OASIS	http://www.oasis-open.org/committees/uddi-spec/
WSDL 1.1	Web Services Description Language	W3C, Mar 2001	http://www.w3.org/tr/wsdl

APPENDIX C: Frequently Asked Questions

NMCI Online FAQs: http://www.eds-gov.com/nmcifaqs/faq_general.asp

TFWeb Online FAQs: https://tfw-opensource.spawar.navy.mil (register and click on FAQs in profile)

The purpose of the appendix is to provide answers to questions that may be common to many developers as they read the updated version of the guidance and begin integrating contents and applications into the NEP. NMCI FAQs are listed at the above URL.

Question: What are the most significant differences between this version and previous versions of the developer's guidance document?

Answer: The information has been updated to focus on conveying just what a developer needs to know in order to integrate a web application, or web service, with the NEP. Unnecessary details that confused this message have been removed.

Question: There are no references to "service modules" in the guidance. What happened to them?

Answer: We have modified our architecture and removed the requirement for service modules. This change has allowed us to simplify the guidance that we provide to developers. Service modules developed under the previous architecture can still be used but will be hosted outside of the immediate portal infrastructure.

Question: Does the removal of the service module requirement change any of the mobile code submission requirements?

Answer: The mobile code submission requirements have not changed. On an as-needed basis, we still approve and host mobile code on our servers in each enclave. The DoD mobile code policy controls the situations in which this solution is applicable.

Question: There are no references to the Enterprise Module Server in the new guidance. What happened to it and what is the impact to developers?

Answer: With the removal of the service module requirement in NEP 2.5, it became unnecessary to describe the Enterprise Module Server and its uses to developers.

Question: What is the difference between a UFS and a portlet?

Answer: A portlet is a visual component that appears in the portal. The UFS is the software component that interfaces with the portal to generate a portlet. The UFS output may be transformed in the portal by 1) applying XSL style sheets to XML documents to produce HTML and/or 2) URL rewriting.

Question: There are no references to "levels of integration" in the new guidance. What happened to them?

Answer: We no longer use the phrase "levels of integration." More importantly, we have separated the discussion of the backend technical implementation requirements from the visual (portlet) aspects. Important parts of fully integrating your application with the NEP are the overall visual appeal, an

integrated look and feel, and the functionality and usefulness of your portlet. There are now three different types of defined portlet integration. It is possible to develop a fully integrated portlet of either type "external content integration" or of type "content integration." Reference portlets can also experience some of these benefits from providing some visual integration with the portal.

Question: a Data Oriented Service What is the difference between a User Facing Service (UFS) and (DOS)?

Answer: A UFS always has an output that can be displayed in a visual format. A UFS also implements the interface with the NEP. A DOS, generally speaking, has data-only output and no visual display formats. Both are recognized as important to implementing an n-tier solution, but a UFS is the only required component for the NEP environment. UFS and DOS components may interact with any number of other UFSs or DOSs, but this is outside the scope of the NEP (migration plans are needed by AMCS representatives).

Question: The guidance seems to concentrate on the interface between the portal and the UFS. There is very little guidance concerning the interface between the UFS and the backend applications. Why?

Answer: We have tried to illustrate some best practices on the backend such as n-tier design and the separation of presentation tier from the business logic and data tiers, but we do not mandate any particular backend system architecture. We also do not require, or suggest, any vendor's products or solutions. We have specified the interoperability standards and interfaces that you need to adhere to. Your application, or web service, should be implemented in a manner that meets your requirements.

Question: My question isn't listed in this FAQ. How can I get it answered?

Answer: You should look at the additional FAQs that can be found on the TFWeb OpSS, located at https://tfw-opensource.spawar.navy.mil. There, you can also post questions to discussion forums or initiate a Technical Assistance Request Form (TARF) for more in-depth assistance.

APPENDIX D: Points of Contact

Table 25: NEADG POCs

Team	POC	Email	Phone
AMTS Team Lead	Bette Fondas	fondasb@spawar.navy.mil	(619) 888-0327
TFWeb Afloat Lead	Terry Howell	thowell@spawar.navy.mil	(619) 553-4111
Deputy TFWeb DC	CAPT Skip Hiser	skip.hiser@navy.mil	(703) 601-4682
PEOIT Developer Guidance Liaison	Steve Parker Temo Villanueva	parkers@saic.com temo@spawar.navy.mil	(703) 868-8334 (858) 826-5168
Commander Task Force Web	Monica Shephard	Monica.Shephard@navy.mil	(757) 836-0548

Table 26: TFWeb AMCS POCs

Command	AMCS POC Code	Phone
Deputy TFWeb Norfolk/AMCS Lead	09WN	(757) 836-3817
BUMED	O9WN3A1	(757) 836-4127
BUPERS	O9WN01	(757) 836-4145
CLF, CPF, CNSL, CNE	09WN3B	(757) 836-4141
CNET	09WN6A1	(757) 836-4114
CNNOC	09WN6B1	(757) 836-4188
CNO	09WN3B	(757) 836-4141
COTF	09WN6B1	(757) 836-4188
CIO	09WN5	(757) 836-4173
METOC	09WN5A	(757) 836-6596
NAVAIR	09WN6	(757) 836-4118
NAVFAC	09WN3A1	(757) 836-4127
NAVSEA	09WN5	(757) 836-4173
NAVSPACE	09WN3A	(757) 836-4113
NAVSUP	09WN3B1	(757) 836-4115
NWC	09WN5B	(757) 836-0099

Command	AMCS POC Code	Phone
NWDC	09WN5B	(757) 836-0099
ONI	09WN6A	(757) 836-4116
ONR	09WN3A1	(757) 836-4127
RESFOR	09WN6A	(757) 836-4116
Safety Center	09WN5B	(757) 836-0099
SPAWAR	09WN 6B	(757) 836-3434
USNO	09WN6B	(757) 836-3434

Legacy Applications POC liaisons

PMO – Brian Barnes barnesbk@spawar.navy.mil 619-524-4557

USMC – Vickie Highlander smblatnmci@mcsc.usmc.mil 703-784-3134

NMCI Help Desk Phone – 1-866-THE NMCI (1-866-843-6624) or helpdesk_sdni@nmci-isf.com

NMCI Help Desk Fax – 1-877-FAX NMCI (1-877-329-6624)

For updates see https://tfw-opensource.spawar.navy.mil (under contacts) and http://www.nmci-isf.com.

DON Applications & Database Management System (DADMS) (https://www.dadms.navy.mil).

APPENDIX E: PRI Data

The PRI data is an XML message that may be optionally transmitted as part of the portal-UFS interface. This message is used as the method of providing portal and user context metadata to the User Facing Service. The portal is unable to insert the PRI Data on any UFS request where it is not in the loop between the portal client and the UFS. To achieve this insertion requires that URLs have been rewritten as specified in APPENDIX H: URL Rewrite Guidelines.

The portal will place the PRI Data in a message called "PRI Data." This optional message insertion is configured in the portal as part of the submission process for each UFS. This message may be sent to the UFS by one of two different methods. For the web application, it is transmitted in a HTTP header variable HTTP_PRIDataRequest. For the web service, it is transmitted as a SOAP header. The service should determine if the PRI Data Request is present, verify that it is valid, and then process it as necessary.

Table 27: PRI Data Element Definition

Data Element Name	Format	Description	Notes
Identity	Complex Element	Encapsulates information pertaining to the portal user's Identity. See below elements.	Contains *CommonIdentity* element and one-to-many *RoleGroup* elements
CommonIdentity	String	The portal user's common identity.	For Example: john.doe
RoleGroup	Complex Element	Contains a RoleType element and a collection of one-to-many Role elements.	Provides information regarding the Portal User's Role Assignments
RoleType	String	Defines the context of the RoleGroup element.	Current Valid Values: Portal
Role	String	Defines a Role assignment in the context of the *RoleGroup* for the portal user	
Device	Complex Element	Encapsulates information pertaining to the client Device which initiated the Request. See below elements.	Contains *DeviceClass*, *DeviceType* and *DeviceVersion* elements
DeviceClass	String	Defines the class of Device that initiated the Request	Current Valid Values: Browser

Data Element Name	Format	Description	Notes
DeviceType	String	Defines the type of Device that initiated the Request in the context of the *DeviceClass*	Current Valid Values: Netscape, IE
DeviceVersion	String	Defines the version relating to the *DeviceType* that initiated the Request	For example: 5.5
Transport	Complex Element	Encapsulates information pertaining to the Transport layer. See below elements.	A UFS may be interested in the Client Node -> Portal Node Transport and the Portal Node -> UFS Node Transport layers
TransportType	String	Defines the context of the Transport.	Current Valid Values: ClientPortal, PortalUFS
IPAddress	String	Contains the IPAddress for the Requesting Node of the Transport	For example: 128.49.196.90
PortalLocation	String	Specifies the location of the Portal in the context of the Transport	Current Valid Values: ashore, afloat
BandwidthConstraints	String	Specifies if there are Bandwidth Constraints in the context of the Transport.	Current Valid Values: true, false
RequestDomain	String	Specifies the domain for the Requesting Node of the Transport	Current Valid Values: mil, com, org, net
TransportProvider	String	Specifies the provider of the Transport layer	Current Valid Values: LAN, wireless
ServiceInstance	Complex Element	Encapsulates information pertaining to the instance of the Service being invoked. See below elements.	Contains *ServiceKey*, *ContentMimeType*, *RewriteURL*, *GenerateCookie*, *InsertStyle*, *RenderXML*, *PortalSessionID and ClientStyle* elements
ServiceKey	String	Contains the bindingKey GUID associated with the Service in the Service Registry	This is the UDDI bindingKey associated with the service in the Service Registry. A 32-digit GUID
ContentMimeType	String	Contains the value for the ContentMimeType t-Model configured for the bindingTemplate in the Service Registry	For example: text/xml, text/html

Data Element Name	Format	Description	Notes
RewriteURL	String	Contains the value for the RewriteURL t-Model configured for the bindingTemplate in the Service Registry	Valid Values: Y, N
GenerateCookie	String	Contains the value for the GenerateCookie t-Model configured for the bindingTemplate in the Service Registry	Valid Values: Y, N
InsertStyle	String	Contains the value for the InsertStyle t-Model configured for the bindingTemplate in the Service Registry	Valid Values: Y, N
RenderXML	String	Contains the value for the RenderXML t-Model configured for the bindingTemplate in the Service Registry	Valid Values: Y, N
PortalSessionID	String	A unique session identifier generated by the Portal product. A 32-digit GUID	The portal dynamically generates and separately maintains this value for each Portlet instance. Applications may use this to maintain state.
ClientStyle	Complex Element	Encapsulates information pertaining to the cascading Stylesheets that a UFS can use to incorporate the Portal look and feel	Contains *TemplateBase* element and one-to-many *StyleGroup* elements
TemplateBase	String	The base URL that contains the set of Portal Cascading Stylesheets	For example: https://portal.tfw.navy.mil /servlet/portal/template/0/ </
StyleGroup	Complex Element	Encapsulates information pertaining to a group of CSS's.	Contains *Content* and *Style* elements
Context	String	Defines the context of the StyleGroup	Valid Values: Browser, IE
Style	String	Defines a specific CSS that the portal has assigned to the UFS based on information from the Request	For example: styles.css

PRI Data XML Schema

The XML schema definition for the PRIRequest document is listed as follows.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
        <xs:element name="PRIRequest">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element name="Identity" type="IdentityElement"/>
                                <xs:element name="Device" type="DeviceElement"/>
                                <xs:element     name="Transport"     type="TransportElement"
maxOccurs="unbounded"/>
                                <xs:element                          name="ServiceInstance"
type="ServiceInstanceElement"/>
                        </xs:sequence>
                </xs:complexType>
        </xs:element>
        <xs:complexType name="IdentityElement">
                <xs:annotation>
                        <xs:documentation>Define        the        Identity        Element
Here</xs:documentation>
                </xs:annotation>
                <xs:sequence>
                        <xs:element name="CommonIdentity" type="xs:string"/>
                        <xs:element name="RoleGroup" type="RoleGroupElement"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="RoleGroupElement">
                <xs:annotation>
                        <xs:documentation> Define RoleGroup Here</xs:documentation>
                </xs:annotation>
                <xs:sequence>
                        <xs:element name="RoleType" type="RoleType"/>
                        <xs:element name="Role" type="Role" maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="DeviceElement">
                <xs:annotation>
                        <xs:documentation>Define the Device Element Here</xs:documentation>
                </xs:annotation>
                <xs:sequence>
                        <xs:element name="DeviceType" type="DeviceType"/>
                        <xs:element name="DeviceVersion" type="DeviceVersion"/>
                        <xs:element name="DeviceClass" type="DeviceClass"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="TransportElement">
                <xs:annotation>
                        <xs:documentation> Define Transport Element Here</xs:documentation>
                </xs:annotation>
                <xs:sequence>
                        <xs:element name="TransportType" type="TransportType"/>
                        <xs:element name="IPAddress" type="IPAddress"/>
                        <xs:element name="PortalLocation" type="PortalLocation"/>
                        <xs:element                              name="BandwidthConstraints"
type="BandwidthConstraints"/>
                        <xs:element name="RequestDomain" type="RequestDomain"/>
                        <xs:element name="TransportProvider" type="TransportProvider"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="ServiceInstanceElement">
                <xs:annotation>
                        <xs:documentation>       Define       ServiceInstance       Element
Here</xs:documentation>
                </xs:annotation>
                <xs:sequence>
                        <xs:element name="ServiceKey" type="ServiceKey"/>
                        <xs:element name="ContentMimeType" type="ContentMimeType"/>
                        <xs:element name="RewriteURL" type="RewriteURL"/>
                        <xs:element name="GenerateCookie" type="GenerateCookie"/>
                        <xs:element name="InsertStyle" type="InsertStyle"/>
                        <xs:element name="RenderXML" type="RenderXML"/>
```

```
                        <xs:element name="PortalSessionID" type="PortalSessionID"/>
                        <xs:element name="ClientStyles" type="ClientStylesElement"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="ClientStylesElement">
                <xs:annotation>
                        <xs:documentation>        Define        ClientStyles        Element
Here</xs:documentation>
                </xs:annotation>
                <xs:sequence>
                        <xs:element name="TemplateBase" type="TemplateBase"/>
                        <xs:element        name="StyleGroup"        type="StyleGroupElement"
maxOccurs="unbounded"/>
                </xs:sequence>
        </xs:complexType>
        <xs:complexType name="StyleGroupElement">
                <xs:annotation>
                        <xs:documentation>         Define         StyleGroup         Element
Here</xs:documentation>
                </xs:annotation>
                <xs:sequence>
                        <xs:element name="Context" type="Context"/>
                        <xs:element name="Style" type="Style"/>
                </xs:sequence>
        </xs:complexType>
        <xs:simpleType name="RoleType">
                <xs:annotation>
                        <xs:documentation>Description of RoleType Here.</xs:documentation>
                </xs:annotation>
                <xs:restriction base="xs:string">
                        <xs:maxLength value="80"/>
                        <xs:enumeration value="Portal"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="Role">
                <xs:annotation>
                        <xs:documentation>Description of Role Here.</xs:documentation>
                </xs:annotation>
                <xs:restriction base="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="DeviceType">
                <xs:annotation>
                        <xs:documentation>Description             of             DeviceType
Here.</xs:documentation>
                </xs:annotation>
                <xs:restriction base="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="DeviceVersion">
                <xs:annotation>
                        <xs:documentation>Description            of            DeviceVersion
Here.</xs:documentation>
                </xs:annotation>
                <xs:restriction base="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="DeviceClass">
                <xs:annotation>
                        <xs:documentation>Description             of             DeviceClass
Here.</xs:documentation>
                </xs:annotation>
                <xs:restriction base="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="TransportType">
                <xs:annotation>
                        <xs:documentation>Description            of            TransportType
Here.</xs:documentation>
                </xs:annotation>
                <xs:restriction base="xs:string">
                        <xs:maxLength value="80" fixed="true"/>
                        <xs:enumeration value="ClientPortal"/>
                        <xs:enumeration value="PortalUFS"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="IPAddress">
                <xs:annotation>
                        <xs:documentation>Description of IPAddress Here.</xs:documentation>
                </xs:annotation>
                <xs:restriction base="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="PortalLocation">
```

```
                <xs:annotation>
                        <xs:documentation>The PortalLocation element will conatin the
location of the portal (either "ashore" or "afloat"). This information was obtained
through the PortalReference object.</xs:documentation>
                </xs:annotation>
                <xs:restriction base="xs:string">
                        <xs:maxLength value="80"/>
                        <xs:enumeration value="ashore"/>
                        <xs:enumeration value="afloat"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="BandwidthConstraints">
                <xs:annotation>
                        <xs:documentation>The BandwidthConstraints element will conatin the
boolean ("True" or "False") value of the flag that informs the service that communication
bandwidth restrictions may exist for this request. This information was returned by the
portal for the specified SessionID.</xs:documentation>
                </xs:annotation>
                <xs:restriction base="xs:string">
                        <xs:enumeration value="true"/>
                        <xs:enumeration value="True"/>
                        <xs:enumeration value="TRUE"/>
                        <xs:enumeration value="false"/>
                        <xs:enumeration value="False"/>
                        <xs:enumeration value="FALSE"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="RequestDomain">
                <xs:annotation>
                        <xs:documentation>Description of RequestDomain
Here.</xs:documentation>
                </xs:annotation>
                <xs:restriction base="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="TransportProvider">
                <xs:annotation>
                        <xs:documentation>Description of TransportProvider
Here.</xs:documentation>
                </xs:annotation>
                <xs:restriction base="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="ServiceKey">
                <xs:annotation>
                        <xs:documentation>Description of ServiceKey
Here.</xs:documentation>
                </xs:annotation>
                <xs:restriction base="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="ContentMimeType">
                <xs:annotation>
                        <xs:documentation>Optional parameter that can help determine how
the PortalConnector should direct the processing of the UFS Response content . Providing
the contentMIMEtype as metadata when registering the service will also allow for faster
processing by the PortalConnector, but will not override the value in the Content-Type
HTTP Header of the UFS Response. </xs:documentation>
                </xs:annotation>
                <xs:restriction base="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="RewriteURL">
                <xs:annotation>
                        <xs:documentation>Controls whether the PortalConnector will attempt
to rewrite URL references in the return stream to proxy all requests back through the
portal. </xs:documentation>
                </xs:annotation>
                <xs:restriction base="xs:string">
                        <xs:enumeration value="Y"/>
                        <xs:enumeration value="N"/>
                        <xs:enumeration value="y"/>
                        <xs:enumeration value="n"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="GenerateCookie">
                <xs:annotation>
                        <xs:documentation>Controls whether the PortalConnector will
generate a session cookie with the information necessary to allow re-invocation. This is
intended for use when a forward proxy is used instead of the URL rewrite
proxy.</xs:documentation>
```

```
                </xs:annotation>
                <xs:restriction base="xs:string">
                        <xs:enumeration value="Y"/>
                        <xs:enumeration value="N"/>
                        <xs:enumeration value="y"/>
                        <xs:enumeration value="n"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="InsertStyle">
                <xs:annotation>
                        <xs:documentation>Inserts the appropriate portal CSS into the HTML
UFS output before sending to the portal client.</xs:documentation>
                </xs:annotation>
                <xs:restriction base="xs:string">
                        <xs:enumeration value="Y"/>
                        <xs:enumeration value="N"/>
                        <xs:enumeration value="y"/>
                        <xs:enumeration value="n"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="RenderXML">
                <xs:annotation>
                        <xs:documentation>Controls whether the PortalConnector will attempt
to render XML using an XSLT Stylesheet reference imbedded in the XML document. Setting
this to N will allow a service to pass the raw XML to the client to support client side
rendering.</xs:documentation>
                </xs:annotation>
                <xs:restriction base="xs:string">
                        <xs:enumeration value="Y"/>
                        <xs:enumeration value="N"/>
                        <xs:enumeration value="y"/>
                        <xs:enumeration value="n"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="PortalSessionID">
                <xs:annotation>
                        <xs:documentation>Description of PortalSessionID Here.  Information
was obtained from the PortalReference Object.</xs:documentation>
                </xs:annotation>
                <xs:restriction base="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="TemplateBase">
                <xs:annotation>
                        <xs:documentation>Description            of            TemplateBase
Here.</xs:documentation>
                </xs:annotation>
                <xs:restriction base="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="Context">
                <xs:annotation>
                        <xs:documentation>Description of Context Here.</xs:documentation>
                </xs:annotation>
                <xs:restriction base="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="Style">
                <xs:annotation>
                        <xs:documentation>Description of Style Here.</xs:documentation>
                </xs:annotation>
                <xs:restriction base="xs:string"/>
        </xs:simpleType>
</xs:schema>
```

PRI Data XML Document Example

An example of a PRIRequest XML document is listed as follows.

```
<?xml version="1.0" encoding="UTF-8"?>
<PRIRequest                          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="https://portal.tfw.navy.mil/schemas/PRIRequest.xsd">
        <Identity>
                <CommonIdentity>john.doe</CommonIdentity>
                <RoleGroup>
                        <RoleType>Portal</RoleType>
                        <Role>ActiveDuty</Role>
                        <Role>JointForces</Role>
                </RoleGroup>
        </Identity>
        <Device>
```

```
                <DeviceType>IE</DeviceType>
                <DeviceVersion>5.5</DeviceVersion>
                <DeviceClass>Browser</DeviceClass>
        </Device>
        <Transport>
                <TransportType>ClientPortal</TransportType>
                <IPAddress>128.49.196.90</IPAddress>
                <PortalLocation>afloat</PortalLocation>
                <BandwidthConstraints>FALSE</BandwidthConstraints>
                <RequestDomain>.mil</RequestDomain>
                <TransportProvider/>
        </Transport>
        <ServiceInstance>
                <ServiceKey>BE22EA89-B18C-49D1-8D54-7DCB4B474F11</ServiceKey>
                <ContentMimeType>text/xml</ContentMimeType>
                <RewriteURL>Y</RewriteURL>
                <GenerateCookie>N</GenerateCookie>
                <InsertStyle>Y</InsertStyle>
                <RenderXML>Y</RenderXML>
                <PortalSessionID>10000000-0100-0000-0002-00000000030F</PortalSessionID>
                <ClientStyles>

        <TemplateBase>https://portal.tfw.navy.mil/servlet/portal/template/0/&lt;/
</TemplateBase>
                        <StyleGroup>
                                <Context>browser</Context>
                                <Style>styles.css</Style>
                        </StyleGroup>
                        <StyleGroup>
                                <Context>IE</Context>
                                <Style>ie_styles.css</Style>
                        </StyleGroup>
                </ClientStyles>
        </ServiceInstance>
</PRIRequest>
```

APPENDIX F: Portal CSS

Introduction

This appendix illustrates the incorporation of the NEP-defined CSS to achieve a uniform look of portal applications and services. For more information on CSS definition, consult the list of references.

Background

The portal has a CSS for each user template defined. The style sheet's elements (referred to as "tags") remain the same between templates. The architecture allows for the portal user to change his or her template, which is equivalent to changing a Windows display scheme. In some instances, maintaining a template's predefined color palette is 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. As more user templates are added, applications and services that use the portal-defined CSS will automatically use those new styles.

Implementation

This section provides the approach to using the portal-defined CSS. A specific example implements the portal-specific cascading style sheet via ASP and JavaScript. Please note that the CSS reference can be inserted automatically by the portal if service metadata option "InsertStyle" is selected (Y). For rapid integration, this approach is recommended over using code to decode the PRI for the purpose of inserting the style sheet reference. Note that the InsertStyle option is only applicable if URLs are being rewritten. See section 3.1.7 Service Registration Metadata for more information about the available service metadata options.

The example below uses client-side and server-side code to decode the PRI and insert the style sheet reference.

Step 1: Get the URL path used to reference the defined CSS.

The PRIRequest message contains the URL reference to the portal's CSS files in <ClientStyle> element. The <templateBase> element contains the fully qualified URL. The <StyleGroup> elements contains the CSS filenames. Notice the context attribute for <style> defines specific browser implementation. Currently, only two are defined: "browser" for all browsers and "IE" for Microsoft® Internet Explorer®.

```
…Other PRI fields…
                <ClientStyles>

        <TemplateBase>https://portal.tfw.navy.mil/servlet/portal/template/0/&lt;/
</TemplateBase>
                        <StyleGroup>
                                <Context>browser</Context>
                                <Style>styles.css</Style>
                        </StyleGroup>
                        <StyleGroup>
                                <Context>IE</Context>
                                <Style>ie_styles.css</Style>
                        </StyleGroup>
                </ClientStyles>
…Other PRI fields…
```

Step 2: Insert path to CSS in HTML stream.

The path to the portal CSS must be included between the <head> and </head> of the HTML document to render correctly. An ASP/Javascript example is shown here., followed by a JSP/JavaScript example, and then by an InsertStyle=Y example. Additional example code is available for download from OpSS.

The following ASP script uses Microsoft XML 3.0 gets and parses a PRI data from the UFS request and inserts the appropriate reference to the CSS in the HTML stream.

```
<%
Dim xmlDOM, TemplateBase, ieStyle, Style, TemplateBaseQry, ieStyleQry, StyleQry

' load pri header content into dom object
Set xmlDOM = CreateObject("MSXML2.DOMDocument.3.0")
xmlDOM.loadXML(HTTP_PRIDataRequest)

' extract TemplateBase element
TemplateBaseQry = ".//TemplateBase"
TemplateBase = xmlDOM.selectsingleNode(TemplateBaseQry).Text
TemplateBase = Replace(TemplateBase,"/</", "/")

' extract Style element with sibling Context element=browser
styleQry = ".//Style[../Context='browser']"
style = xmlDOM.selectsingleNode(styleQry).Text

' extract Style element with sibling Context element=IE
ieStyleQry = ".//Style[../Context='IE']"
ieStyle = xmlDOM.selectsingleNode(ieStyleQry).Text

Set xmlDOM = Nothing
%>


<html>
<head>
<title>Hello</title>
<LINK HREF='<%= TemplateBase & Style %>' REL='stylesheet' TYPE='text/css'>
<script language="JavaScript">
<!--
if( navigator.appName == 'Microsoft Internet Explorer' )
{
  document.write("<LINK   HREF='<%   =   TemplateBase   &   ieStyle   %>'   REL='stylesheet'
TYPE='text/css'>");
}
-->
</script>
</head>
<body>
<h3>Hello, World</h3>
</body>
</html>
```

Which gives the following HTML and JavaScript to the browser.

```
<html>
<head>
<title>Hello</title>
<LINK              HREF='https://portal.tfw.navy.mil/servlet/portal/template/0/styles.css'
REL='stylesheet' TYPE='text/css'>
<script language="JavaScript">
<!--
if( navigator.appName == 'Microsoft Internet Explorer' )
{
  document.write("<LINK
HREF='https://portal.tfw.navy.mil/servlet/portal/template/0/ie_styles.css'
REL='stylesheet' TYPE='text/css'>");
}
-->
</script>
```

```
</head>
<body>
<h3>Hello, World</h3>
</body>
</html>
```

The above client-side JavaScript code allows the Internet Explorer (IE) specific CSS to be included if the user's browser is determined to be IE. This can also be done with server-side code.

Following is a JSP implementation with the same features shown in the ASP example above.

```
<%@ page import="org.apache.xerces.parsers.DOMParser"%>
<%@ page import="org.xml.sax.*"%>
<%@ page import="java.io.*"%>
<%@ page import="org.apache.xpath.*"%>
<%@ page import="org.w3c.dom.Element"%>


<%
// Read PRI content into parser
StringReader sr = new StringReader(request.getHeader("HTTP_PRIDataRequest");
InputSource is = new InputSource(sr);
DOMParser parser = new DOMParser();
parser.parse(is);
Element root = parser.getDocument().getDocumentElement();

// Extract TemplateBase element
String TemplateBaseQry = ".//TemplateBase/text()";
String TemplateBase = XPathAPI.selectSingleNode(root,TemplateBaseQry).getNodeValue();
if(TemplateBase.endsWith("</"))
TemplateBase=TemplateBase.substring(0,TemplateBase.length()-2);

// Extract Style element with sibling Context element=browser
String StyleQry = ".//Style[../Context='browser']/text()";
String Style = XPathAPI.selectSingleNode(root,StyleQry).getNodeValue();

// Extract Style element with sibling Context element=IE
String ieStyleQry = ".//Style[../Context='IE']/text()";
String ieStyle = XPathAPI.selectSingleNode(root,ieStyleQry).getNodeValue();
%>



<html>
<head>
<title>Hello</title>
<LINK HREF='<%= TemplateBase + Style %>' REL='stylesheet' TYPE='text/css'>
<script language="JavaScript">
<!--
if( navigator.appName == 'Microsoft Internet Explorer' )
{
  document.write("<LINK   HREF='<%=   TemplateBase   +   ieStyle   %>'   REL='stylesheet'
TYPE='text/css'>");
}
-->
</script>
</head>
<body>
<h3>Hello, World</h3>
</body>
<html>
```

Following is an example where the InsertStyle=Y metadata option was selected. Note that there is nothing that the developer needs to add to the code to use this feature, however, there must exist a <head> section. The developer also selected metadata option Rewrite URLs =Y to enable the portal to process the content for CSS insertion on all UFS requests.

```
<html>
<head>
<title>Hello</title>
</head>
<body>
<h3>Hello, World</h3>
</body>
<html>
```

When processed by the portal this gives the following HTML to the browser.

```
<html>
<head>
<title>Hello</title>
```

```
<LINK              HREF='https://portal.tfw.navy.mil/servlet/portal/template/0/styles.css'
REL='stylesheet' TYPE='text/css'>
<LINK          HREF='https://portal.tfw.navy.mil/servlet/portal/template/0/ie_styles.css'
REL='stylesheet' TYPE='text/css'>
</head>
<body>
<h3>Hello, World</h3>
</body>
</html>
```

Step 3: Remove all HTML font modification attributes/tags to allow the CSS to render the defined style.

Any font modification will cause the rendered page's look to be different than the portal's implementation. The following example HTML 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>
```

With the CSS referenced as in Step 2, the above HTML code would remove all of its font attributes, such as, font size="10" type="Arial" color="blue. The cleaned up code would now look like this:

```
<Table>
      <tr>
<td><b>Welcome, John Doe!</b></td>
      </tr>
</Table>
```

With no Font attributes defined, the style sheet will control the output:

The <td> tag will be used in this example to define the font attributes.

As seen in the Style Sheet:

```
td {
FONT 8pt univers, verdana, arial, Helvetica, sans-serif; color: #00066
}
```

Step 4: Add the defined CSS tags to document if needed.

For all styles in the CSS not assigned to HTML tags, a specific attribute must be added to the HTML tag. A list of available CSS tags defined by the portal CSS is listed at the end of this appendix. Use of the additional tags is usually not necessary. The following HTML illustrates the use of a style defined in the portal CSS.

```
<body>Normal Text<br/>
<p class="folderselected">This is in the folderselected style</p>
Normal text<br/>
</body>
```

Defined Classes

Table 28: SS Tag Descriptions for Style Sheets 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 in the following text are one example of a template available on the portal. Figure 23: Application Integration & Testing Lab (AIT) Process through Figure 26: Sample Screen 3 with Style Tags map out where some of these tags have been used in a template for the portal. These may be used as guidelines for service developers to reference to prevent style tag collision when adding the class names to their web applications. In addition, a developer may download CSS files for testing from the Open Source site.

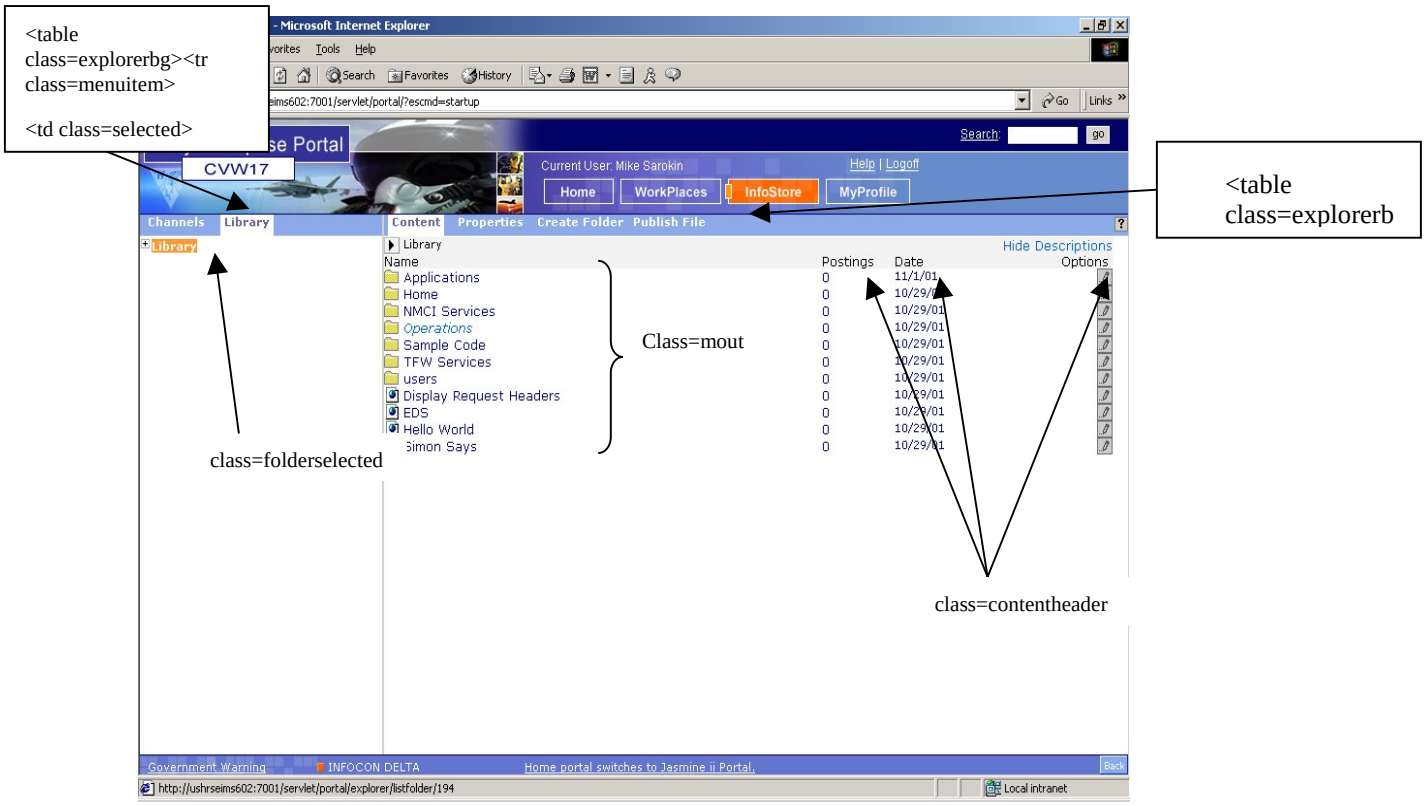
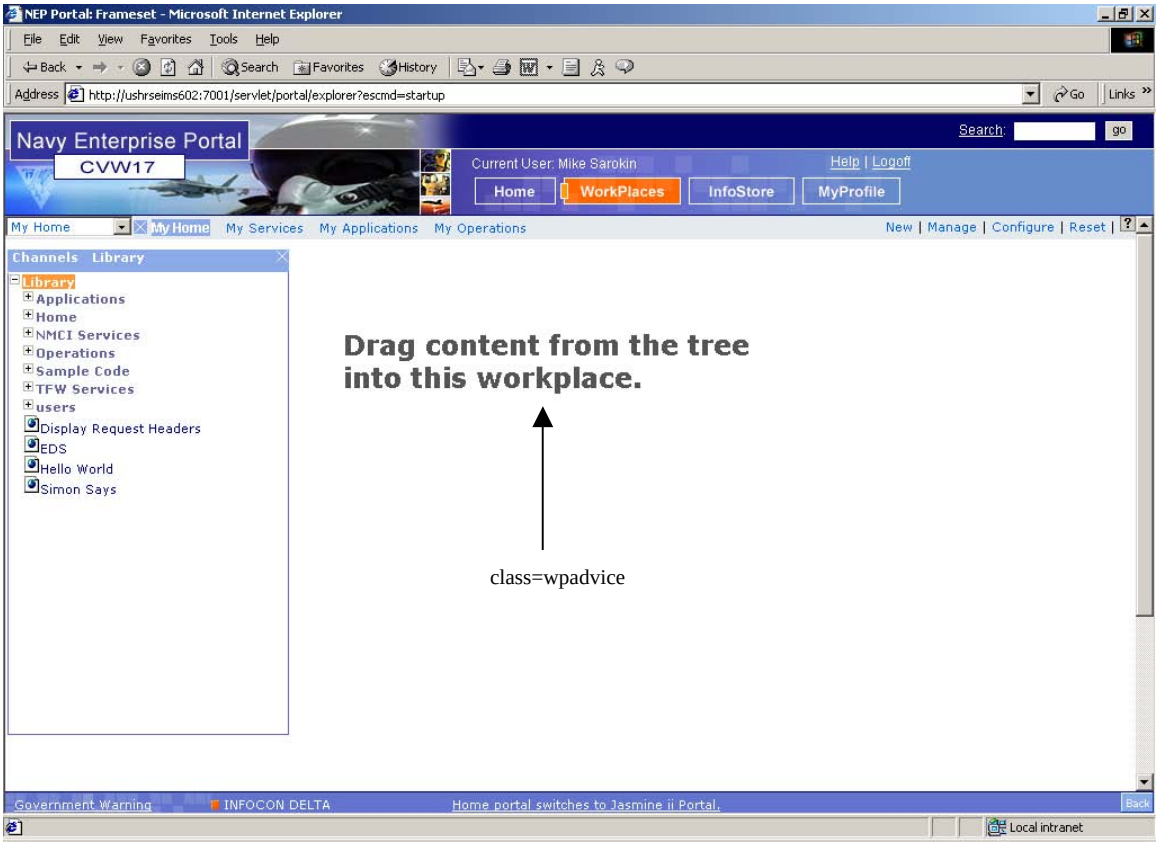
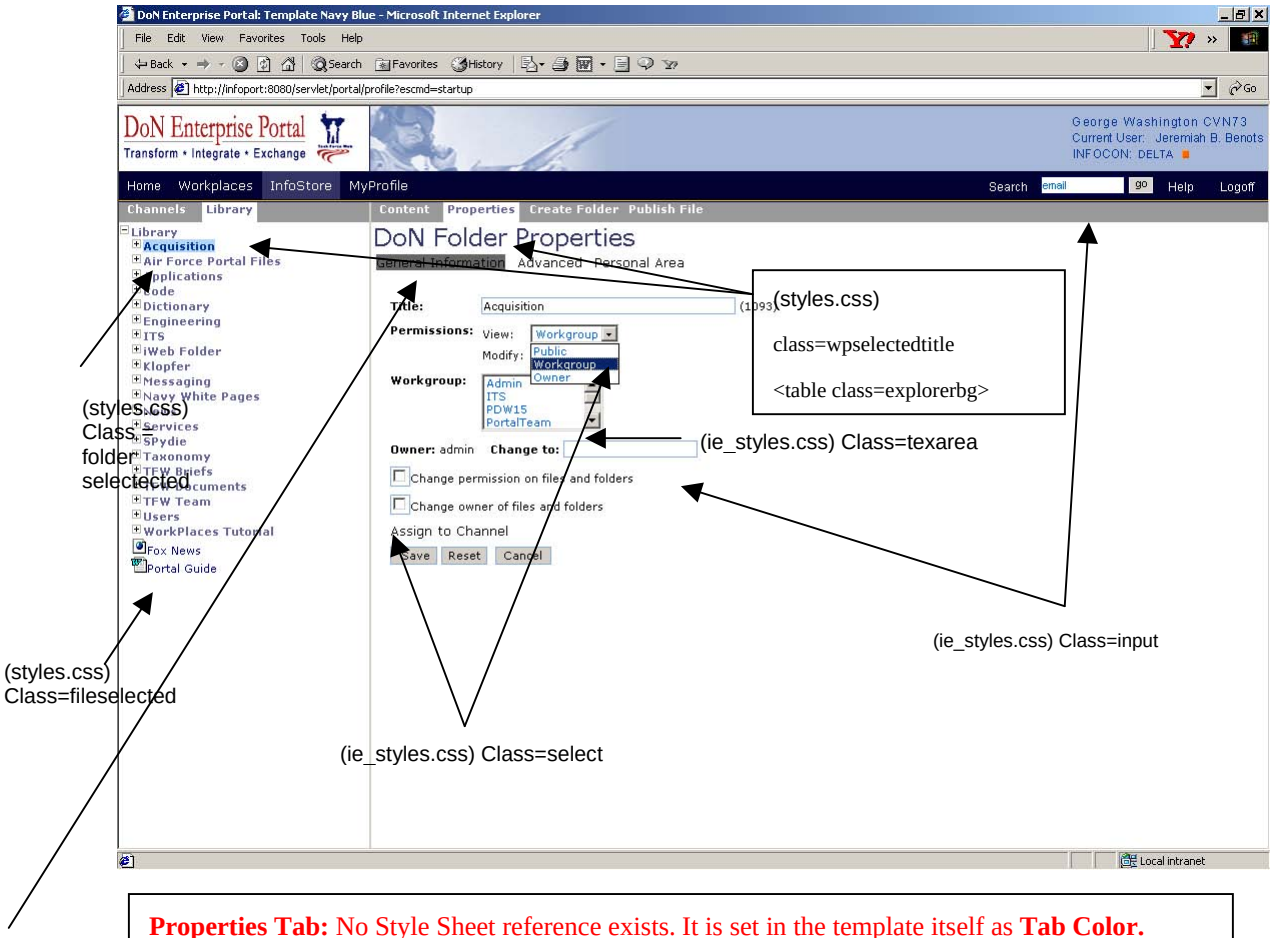


Figure 24: Sample Screen 1 with Style Tags



NOTE: Some Portlet element attributes, such as the portlet border color, are not in the CSS, but are controlled by Administrators under the Workplace Configuration page, under Template Administration.

Figure 25: Sample Screen 2 with Style Tags



Properties Tab: No Style Sheet reference exists. It is set in the template itself as **Tab Color.**

Figure 26: Sample Screen 3 with Style Tags

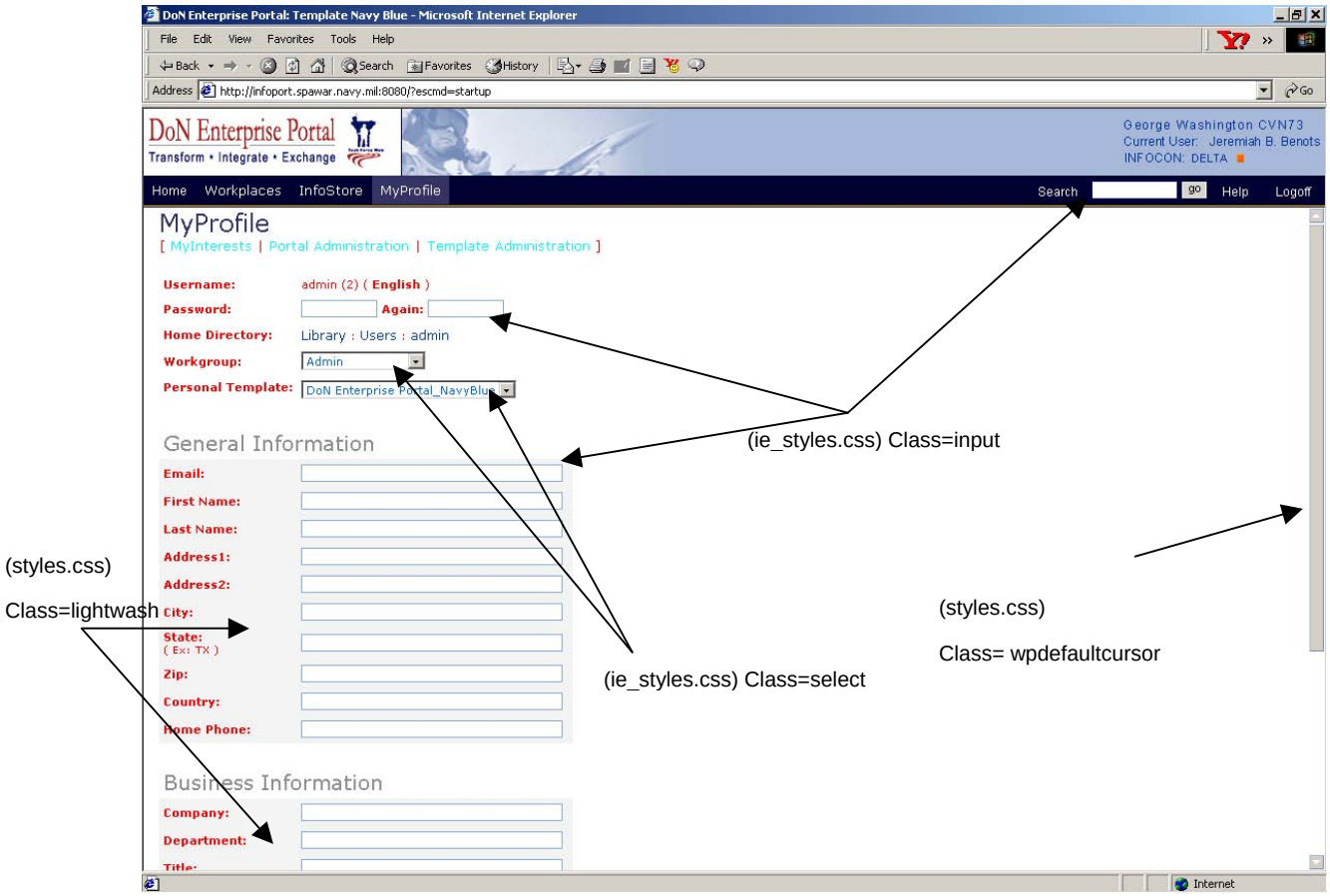


Figure 27: Sample Screen 4 with Style Tags

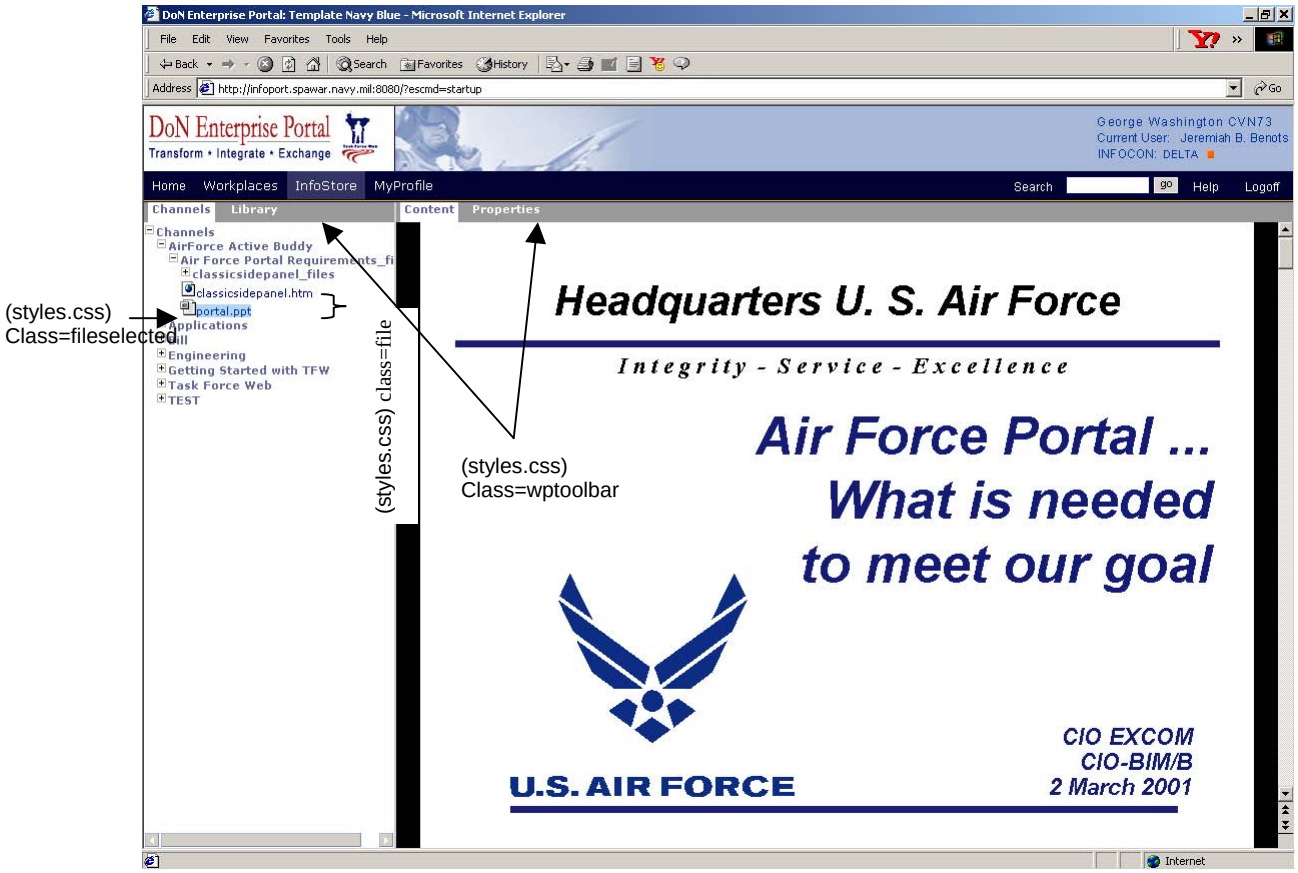


Figure 28: Sample Screen 5 with Style Tags

Table 28: SS Tag Descriptions for Style Sheets

Element/Class	Description	How to reference
A	Hyperlink	No reference needed
Td	**Table Data**	No reference needed
Td.footnote	Footnote attribute (found on Local page)	Class="footnote"
Th	**Table Header**	No reference needed
Th.footnote	Footnote attribute (found on Local page)	Class="footnote"
Contentheader	Header	class="contentheader"
currentdirectory	Current Directory	class="currentdirectory"

Element/Class	Description	How to reference
Explorerbg	Background Color	class="explorerbg"
explorertabindicator	Explorer Tab Indicator	class="explorertabindicator"
Explorertablebg		class="explorertablebg"
File	File font	class="file"
Fileselected	Selected File	class="fileselected"
Folder	**Folder Name**	class="folder"
Folderselected	**Selected Folder**	Class="folderselected"
font1	Font option	class="font1"
font2	Font option	class="font2"
font3	Font option	class="font3"
Libraryselected	Selected Library	class="libraryselected"
Librarypath	Background color option	class="librarypath"
Lightwash	Background color option	class="lightwash"
Mediumwash	Background color option	class="mediumwash"
Menuitem	Menu Items	class="menuitem"
Menulink	Menu Link	class="menulink"
Message		class="message"
Mout	Mouse Out	class="mout"
Mover	Mouse Over	class="mover"
Na		class="na"
nc1		class="nc1"
nc2		class="nc2"
Nh		class="nh"
Notselected		class="notselected"
Selected	**Selected Option**	class="selected"
Title	Title	class="title"
Toolbar		class="toolbar"
Upload		class="upload"
White		class="white"
Wpadvice	Large Instructions	class="wpadvice"
wpcontentlist1		class="wpcontentlist1"
wpcontentlist2		class="wpcontentlist2"
Wpdefaultcursor	Default cursor style	class="wpdefaultcursor"

Element/Class	Description	How to reference
Wpelemtoolbar		class="wpelemtoolbar"
Wpoptions	Options	class="wpoptions"
Wpselectedtitle	Selected title	class="wpselectedtitle"
Wptitle		class="wptitle"
wptoolbar	Toolbar	class="wptoolbar"
Wptreetop	Background image	class="wptreetop"

ID as selector	Description	How to reference
#ChngMod		Id="chngMod"
#command		Id="command"
#hdrSubTitle2b		Id="hdrSubTitle2b"
#hdrSubTitle2c		Id="hdrSubTitle2c"
#INFOCON		Id="INFOCON"
#Ln		Id="Ln"
#nav		Id="nav"
#secLinks		Id=" secLinks"

Ie_style.css Elements

class as selector	Description	How to reference
input		No reference needed
select		No reference needed
Textarea		No reference needed
ID as selector	**Description**	**How to reference**
#button	Used for form buttons, e.g., submit, save, etc. Differentiate between "input" form elements.	Id="button"

APPENDIX G: Application Security

Introduction

There are two important interfaces that must be secured when migrating applications and content into the NEP: the portlet interface and the UFS interface to the portal. It is the responsibility of the NEP to secure the portlet interface. It is the responsibility of the application owner to secure the UFS interface, with the assistance of the services provided by the NEP. Furthermore, the application owner must address application-specific security measures for the UFS (and the hardware supporting it) to meet the requirements of the application's developmental DAA.

The purpose of this appendix is to provide detailed information about the security mechanisms and options that are provided by the portal. It does not attempt to address application-specific security mechanisms. Developers and integrators should become familiar with this appendix in order to select the most appropriate portal-provided security mechanisms to support the integration of their applications. Figure 29: Scope of Application Security Appendix explains this appendix's scope.

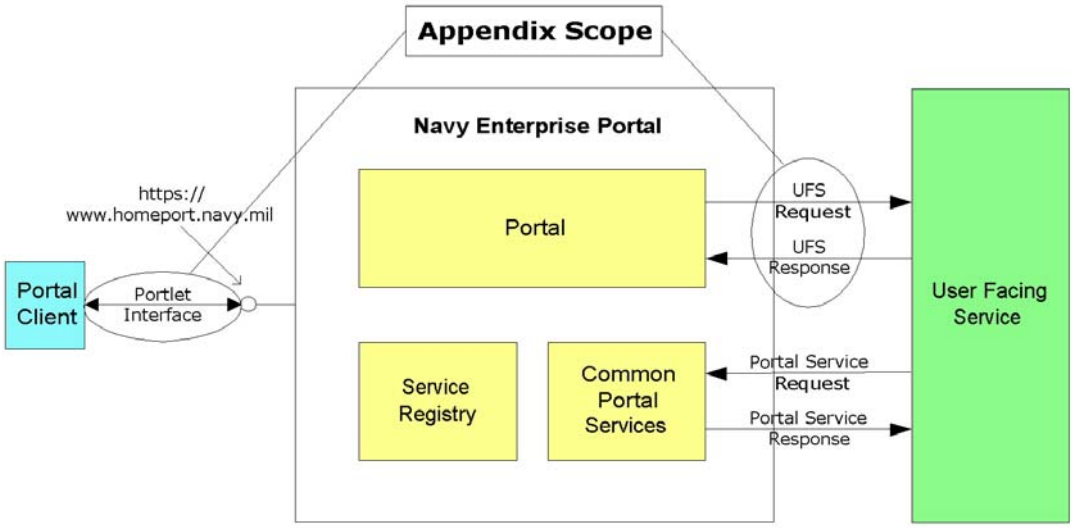


Figure 29: Scope of Application Security Appendix

The most significant decision that must be made by the application owner is whether the security used by the UFS interface to the portal provides an adequate level of trust for the data accessed through the UFS. If the application owner chooses not to trust these mechanisms, the application owner may choose to require a separate authentication to the application. If the application re-authenticates portal users, it must handle the authentication in a portal-friendly manner (such as not automatically popping up new windows).

Portlet Interface Security Description

Current Architecture

Note: The NEP does not support all certificates issued by DoD Certificate Authorities (CAs). Currently, only certificates in the hierarchy based on the DoD Class 3 Root CA certificate are supported. (This includes the following certificates: DoD Class 3 CA-3, DoD Class 3 CA-4, and DoD Class 3 CAC CA.) Certificates in the hierarchy based on the DoD PKI Med Root CA are not supported. (This includes the Med CA-1 and Med CA-2 certificates.)

The portal performs a two-factor authentication:

A portal client is asked to present a login and password.

Encrypted Connection – the portal can interface with the application using an encrypted connection. Currently, the NEP can only support SSL connections over the UFS interface. Other encryption standards such as TLS are not yet supported.

The certificates are checked against the appropriate DoD root trust chain and Certificate Revocation List (CRL) to ensure they are valid certificates.

The user enters his or her Common Identity UserID and password.
The UserID is authenticated against the Naval Global Directory Service (NGDS) and user roles are passed to the portal. User roles determine a specific library of services that are made available to the user.

After authentication, the portal can then provide the Common Identity UserID to any portal service via HTTP or HTTPS by the PRI Request and/or authorization HTTP header field.

Objective Architecture

A COTS single sign on (SSO) product is the next step in the access security implementation for services integrated into the NEP. Presently, a product has not been implemented for use. Guidance for the implementation of the SSO product will be provided in the future. Contact the AMCS for further information.

UFS Interface Security Implementation

The NEP can provide two items for applications and services via the UFS interface:

Encrypted Connection – the portal can interface to the application using an encrypted connection. Currently, NEP policy states that the portal shall only support SSL connections to UFS or DOS.

Common Identity – the portal sends the common identity through the PRI Request and/or authorization header field (recommended) in the HTTP header.

The multiple combinations of the above items are listed in

Table 29: Security Implementation Combinations.

Table 29: Security Implementation Combinations

Common Identity Sent	Encrypted Connection (SSL)
No	No
Yes	No
No	Yes
Yes	Yes

The following examples each use a different combination of the capabilities of the portal to secure the UFS interface. These examples are not all-inclusive. Each example has advantages and disadvantages and differing levels of trust. In addition, an example using the future SSO product is included. The decision on which combination to use is based on the information needed within the application/service. Section V of *Web Site Administration Policies & Procedures, November 25, 1998,* available on http://www.defenselink.mil/webmasters, contains examples and best practices for information security for web sites.

Example 1: A General Public Service

Portal Actions: The portal provides no common identity information or PRI Request header, nor is the UFS interface encrypted.

Examples: Early Bird News Service, National Weather Service information.

Table 30: Example 1: Comparison

Advantages	Disadvantages
Higher Performance	Higher risks since host server is exposed to the general Internet population, however equal to current risk assumed.
No application userID/password infrastructure required	Should not be used for applications requiring any form of authentication/authorization.
Easy to implement	

Level of Trust: Users of a service using this security methodology are considered to be "anonymous" because they cannot be authenticated over the unencrypted UFS interface. Data passed over the UFS interface will also be in the clear. This methodology provides no trust.

Example 2: An SSL Service

Portal Actions: The portal provides no common identity information, or PRI Request header, however the UFS interface is encrypted.

Note: Applications and services that are designed not to be accessible by the general public must obtain a DoD server certificate and are required to use two-way SSL per CNO message DTG 301704Z NOV 00.

Example: https://www.infosec.navy.mil

Table 31: Example 2: Comparison

Advantages	Disadvantages
Most browsers support SSL.	Requires a DoD PKI server certificate.
Existing means of application authentication can still be used.	Lower performance.

Level of Trust: To provide trust in the user's identity, the service must re-authenticate the user independently of the portal (in a portal-friendly manner). Data sent over the UFS interface will also be encrypted, providing trust that it will not be compromised in transit. With user re-authentication, this methodology provides a high level of trust. Without authentication, this methodology provides no trust of the user's identity, but a high level of trust for the data.

Example 3: Portal-supplied Common Identity without SSL

Portal Actions: The portal provides common identity information; however, the UFS interface is not encrypted.

Note: The application/service can use the common identity as a means of identifying users, and possibly tailor its functionality based on the common identity. For example, an application or service receives the common identity for tracking information such as date of last login. Implementing the common identity on existing applications will require a mapping from the Common Identity to existing application userID. Authentication and authorization of application rights without SSL is prohibited by current policy.

Table 32: Example 3: Comparison

Advantages	Disadvantages:
Common Identity is available to provide personalization of the service.	The existing applications will need modifications to support the common identity (Mapping).
Higher performance without SSL.	Must read header field or parse PRI to determine common identity.
	Should not be used for applications requiring any form of authentication/authorization.

Level of Trust: Users of a service using this security methodology are considered to be "anonymous" because they cannot be authenticated over the unencrypted UFS Interface. However, if the application owner and the DAA accept the risk, the Common Identity could be used for personalization purposes. Data passed over the UFS Interface will also be in the clear. This methodology provides a very low level of trust.

Example 4: Portal-supplied Common Identity with SSL

Portal Actions: The portal provides Common Identity information, and the UFS interface is encrypted.

The application/service uses the common identity as a means of identifying users, tailors its functionality, and possibly assigns application local roles to those users. A use of this combination would be to mimic a SSO capability. An application/service may choose to accept the passed common identity to allow access and perform authorization for that user.

Table 33: Example 4: Comparison

Advantages	Disadvantages
Support for the user's identity is now shifted away from the application/service developer. The application owner no longer needs to manage user passwords, but must still manage users for means of authorization.	The existing applications will need modifications to support the common identity. Application local user information will need to be stored in a local database.
Common identity may be used for re-authentication to the application/service, because passwords are sent encrypted.	Requires a DoD server certificate.
Can eliminate multiple login screens.	Lower performance.

Level of Trust: The service may choose to use the common identity (without password) as its authentication or may require the user to re-authenticate to an internal user database or the NGDS common identity store. Data sent over the UFS interface will also be encrypted providing trust that it will not be compromised in transit. Using the Common Identity (without password) as user authentication, this methodology provides a medium level of trust. With user re-authentication (against internal store or NGDS), this methodology provides a high level of trust.

Example 5: Portal-supplied Common Identity with COTS Single Sign On (SSO) product and SSL

Portal Actions: The portal provides Common Identity information, and the UFS interface is encrypted.

Level of Trust: The service uses the common identity as its authentication, through the SSO product. Data sent over the UFS interface will also be encrypted, providing trust that it will not be compromised in transit. This methodology provides a high level of trust.

Table 34: Example 5: Comparison

Advantages	Disadvantages
Support for the user identity is now shifted away from the application/service developer. The application owner no longer needs to manage user passwords, but must still manage users for means of authorization.	Existing applications have to map the common identity to the existing user names. Application local user information will need to be stored in a local database.
Passwords are never sent over the UFS interface.	Requires a DoD server certificate.
Uses the portal login for application authentication. Eliminates multiple login screens.	Lower performance.

Level of Trust: The service uses the common identity as its authentication, through the SSO product. Data sent over the UFS interface will also be encrypted providing trust that it will not be compromised in transit. This methodology provides a high level of trust.

APPENDIX H: URL Rewrite Guidelines

The portal may optionally be configured to act as a proxy between the portal client and the UFSs. This configuration is required for those services that:

- Need to be available to users within enclaves where firewalls may block direct access to the UFS. The portal proxies each request to the UFS and then passes the UFS response back to the portal client.
- Need to receive the PRI Data (portal and user context metadata) in the UFS request. See section 2.1.8.3.2 UFS Request, and section 3.1.7 Service Registration Metadata, and APPENDIX E: PRI Data for more information.
- Need to receive the Common Identity of the user in the UFS request. See section 2.1.8.3.2 UFS Request, and section 3.1.7 Service Registration Metadata for more information about the Common Identity.
- Need to use the automatic insertion of the Portal CSS URL reference into the UFS response HTML stream. See section 2.1.7.4.3
- Portlet Service Responsibilities, and section 3.1.7 Service Registration Metadata, and APPENDIX F: Portal CSS for more information.
- Need to use the portal XML rendering into HTML using XSLT stylesheets in the UFS response stream. See section 2.1.7.4.3
- Portlet Service Responsibilities, and section 3.1.7 Service Registration Metadata for more information.

When the portal is used to proxy access to web content, it does so by re-writing the URL links to redirect connections back through the portal. For the HTML content in the UFS response, the portal examines the HTML and looks for certain key URL tags. When it encounters one of these tags, it prepends a URL reference to the portal in front of the original UFS URL. When the portal client transmits a reference of this type to the portal, the portal sets up an HTTP client session and requests the UFS content on behalf of the portal client. The returned UFS response stream is then examined for URLs to re-write and is returned to the portal client.

There are two different techniques that can be used by the UFS to accomplish properly rewritten URLs.

1. By the portal in the UFS response. See the flow diagram (Figure 16: Portal Response Processing) to see where the URL Rewrite processing takes place within the overall portal response processing sequence. This method requires that the UFS use one of the following:

Select the service metadata option “Rewrite URLs = Yes”. See section 3.1.7 Service Registration Metadata for more information about this option. This option is recommended.
Use the Portlet Service Request query string parameter “rewriteURL=Y”. See section 2.1.7.4.2 Portlet Service Request for more information about this parameter.

2. By the UFS as an independent call to the URL Rewrite Service. See section 2.1.9.2.2 URL Rewrite Interface for more information about this service.

The portal identifies the following HTML tags for re-writing:

```
HREF=
SRC=
```

```
URL=
BACKGROUND=
ACTION=
```

All other methods for producing links, especially those that rely on dynamic client side code (such as JavaScript) or code embedded in objects, are not supported. The portal cannot handle links it can't find to re-write.

Rewritten URLs Example

As an example, assume that the following HTML source file is located at URL http://myapp.navy.mil/info/MyFile.html and has been retrieved by the portal.

BEFORE:

```
<HEAD>
        <TITLE>Page Title</TITLE>
</HEAD>
<body>
        <img src = "http://myapp.navy.mil/images/mygif.gif">

        <img src = "/images/mygif2.gif">

        <a href = "nextpage.html">Next Page</a>
</body>
```

After URLs have been rewritten by the portal, the HTML document would look something like this:

AFTER:

```
<HEAD>
        <TITLE>Page Title</TITLE>
</HEAD>

<body>

<img                                        src                                        =
"https://portal.tfw.navy.mil/nep/PortalConnector/user=joe@http://myapp.navy.mil/images/my
gif.gif">

<img                                        src                                        =
"https://portal.tfw.navy.mil/nep/PortalConnector/user=joe@http://myapp.navy.mil/images/my
gif2.gif">

<a                                        href                                        =
"https://portal.tfw.navy.mil/nep/PortalConnector/user=joe@http://myapp.navy.mil/info/next
page.html"> Next Page</a>

</body>
```

Examples of URLs that can't be rewritten

Relative URLs to Remote Servers

In certain specific cases of content with relative URLs, the portal will not know the fully qualified URL. As defined in the HTML standard, an undefined or underivable base will result in an unresolvable URL and a "broken" link.

As an example, this content may have passed through the UFS from other DOSs. In this case, the use of the HTML header tag "BASE HREF=" may be required. Refer to the example as follows.

```
<HEAD>
        <TITLE>Page Title</TITLE>
        <BASE HREF="http://MyDataOrientedService.navy.mil/">
</HEAD>
<body>
        <img src = "/images/mygif2.gif">
</body>
```

This allows the portal 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

An alternate method to resolve this issue is to use fully qualified addresses on all references.

```
<HEAD>
        <TITLE>Page Title</TITLE>
</HEAD>
<body>
        <img src= "http://MyDataOrientedService.navy.mil/images/mygif2.gif">
</body>
```

Examples of URLs that can't be rewritten

Key – red = will not proxy

– green = will proxy

```
<HEAD>
        <TITLE>Page Title</TITLE>
</HEAD>

<script language="whateverscript">
        function MouseOver {
                document.imagename.src = "/images/mover.gif"
        }
        function MouseOut {
                document.imagename.src = "/images/mout.gif"
        }
</script>

<body>

        <a href="positions.html" target="homepage"

onMouseOver='document.images["jobs"].src="images/buttons/jobbtn-over.gif"'

onMouseOut='document.images["jobs"].src="images/buttons/jobbtn.gif"'
>
<a href="#">top</a>

<form action="mailto:jobinfo@spawar.navy.mil">

<img src="">

<p>Some sample text: src = "images/buttons/sample.gif"</p>

</body>
```

URLs are not proxied if

Embedded between two script tags (scripting language is irrelevant: vbscript, jscript, javascript)
Embedded within an event (onMouseOver, OnLoad, etc…)
If the tag is empty

If it is part of a request parameter (href=”http://www.spawar.navy.mil?action=findjobs”). The href will be proxied, but the action will not
If the url is not found within a “<” and “>”
Any value starting with “javascript:”, “vbscript:”, or “mailto:”

APPENDIX I: Functional Area Management (FAM) SECNAV Reference

THE UNDER SECRETARY OF THE NAVY
WASHINGTON, D.C. 20350-1000

14 May 2002

MEMORANDUM FOR DISTRIBUTION

Subj: DESIGNATION OF DEPARTMENT OF NAVY (DON) FUNCTIONAL AREA MANAGERS

Ref: (a) CNO WASHINGTON DC 252250Z FEB 02
(b) SECDEF memo of 19 Jul 01
(c) USD(C) memo of 12 Oct 2001

Encl: (1) DON Applications and Database Management Process
(2) DON Functional Areas and Functional Area Managers

Implementation of the Navy Marine Corps Intranet (NMCI) has identified numerous duplicative Information Technology (IT) applications and databases. Significantly reducing the number of DON IT applications and databases reduces costs and also facilitates implementation of NMCI, the Defense Financial Management Modernization Program (DFMMP), web enablement, eBusiness strategies and the use of common business and administrative processes across the Department. There are both a short term and long term aspects to this effort.

Recently, Navy Echelon II Commanders were directed by reference (a) to implement processes to reduce IT applications and databases within their commands. The Marine Corps has already undertaken a major effort to reduce USMC IT applications and databases. IT application reduction goals and metrics will be established shortly by the Service Chiefs, and these metrics will be compiled and reported to me on a biweekly basis by DON CIO, with copies to the General Counsel, all Assistant Secretaries of the Navy, and the Service Chiefs.

The next step is for the responsible organizations within each functional area to reduce their IT applications and databases using the framework shown in enclosure (1). The responsible organizations, as detailed in enclosure (2), in consultation with the key Secretariat stakeholders, shall designate Functional Area Managers (FAMs) for each of the listed functional areas. A single FAM for each functional area is preferred, but where appropriate, a FAM for both the Navy and Marine Corps may be appointed, after consultation with the key stakeholders.

Subj: DESIGNATION OF DEPARTMENT OF NAVY (DON) FUNCTIONAL AREA MANAGERS

The FAMs, within their functional areas, shall: be responsible and accountable for overseeing the reduction and consolidation of IT applications and databases; have the authority to direct migration, consolidation, or retirement of applications and databases; develop and manage IT applications and database portfolios; ensure that technology strategies are aligned with the business and administration processes and warfighting strategies; work closely with the DON CIO and the DON Information Executive Committee Service representatives to ensure that standardized DON processes and procedures are consistently used to accomplish this task; coordinate their actions with the appropriate OSD Principal Staff Assistants for Joint applications, DFMMP Manager for financial and business systems, Director, NMCI, the Echelon II Navy commands, and the Major USMC commands; and consult with the key secretariat stakeholders as needed and keep them informed of progress. Functional Data Managers (FDMs) are responsible for reducing the databases in each of these same functional areas. The FDMs shall work closely with and report to the FAMs.

The FAMs will meet monthly to develop and coordinate application management strategies and implementation plans and to share lessons learned and best practices. The meeting will be co-chaired by the DON CIO and the Director, Navy Staff.

Reference (b) established the DFMMP and reference (c) requires USD(C) approval to change financial systems and their related non-financial (feeder) systems. Since the DFMMP will impact many applications and databases in multiple functional areas, it is imperative that the FAMs responsible for business systems migrate, consolidate and retire their applications and databases in accordance with DFMMP and ASN(FM&C) direction and guidance. Additionally, FAMs will coordinate with ASN (FM&C) to ensure that DFMMP's DoD-wide architecture supports DON requirements.

Request that addressees designate FAMs, as specified in enclosure (2), to the DON CIO within 5 working days of this memorandum. Please forward the names to Mr. Brian Wilczynski, at wilczynski.brian@hq.navy.mil. He can also be reached at 703-607-5653.

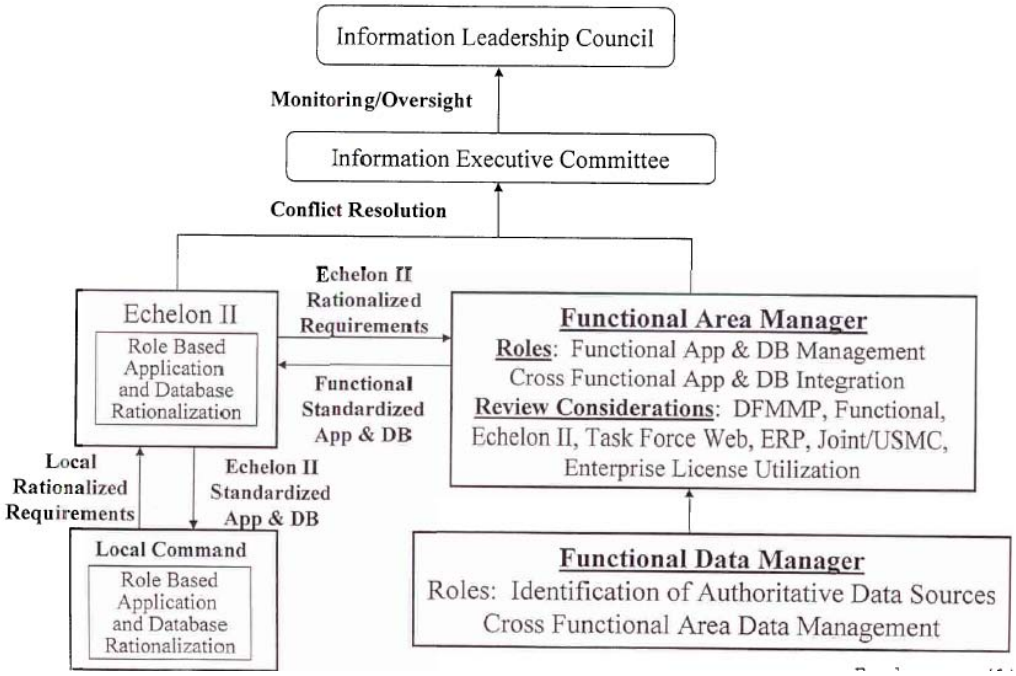
Susan Morrisey Livingstone

Subj: DESIGNATION OF DEPARTMENT OF NAVY (DON) FUNCTIONAL AREA MANAGERS

Distribution:
VCNO
ACMC
ASN (FM&C)
ASN (RD&A)
ASN (M&RA)
ASN (I&E)
OGC
JAG

Copy to:
SECNAV
AAUSN
DNS
DMCS

DON Applications and Database Management Framework



DON FUNCTIONAL AREAS AND FUNCTIONAL AREA MANAGERS

Functional Area	Responsible Organization	Key Secretariat Stakeholders	Functional Area Manager (Designated by Responsible Organization)	Optional USMC Functional Area Manager
Acquisition	ASN(RD&A)			XXXXXXXXXXXXXX
Financial Management	ASN(FM&C)			XXXXXXXXXXXXXX
Civilian Personnel	ASN(M&RA)			XXXXXXXXXXXXXX
Legal	GC/JAG			XXXXXXXXXXXXXX
Administration	DNS; HQMC AR	AAUSN		
Manpower and Personnel	OPNAV N1; HQMC M&RA	ASN(M&RA)		
Intelligence and Cryptology	OPNAV N2; HQMC I	ASN(RD&A)		
Logistics (includes Facilities Mgmt, Environment)	OPNAV N4; HQMC I&L	ASN(I&E); ASN(RD&A)		
Readiness	OPNAV N4; HQMC PP&O	ASN(RD&A)		
Command, Control and Communications	OPNAV N6/N7; HQMC C4	ASN(RD&A)		
Information Warfare	OPNAV N6/N7; HQMC PP&O	ASN(RD&A)		
Modeling and Simulation	OPNAV N6/N7; MCSC SE&I	ASN(RD&A)		
Weapons Planning and Control	OPNAV N6/N7; HQMC	ASN(RD&A)		
Training and Education	OPNAV N79; TECOM	ASN(MR&A)		
Resources, Requirements, and Assessments	OPNAV N8; HQMC P&R	ASN(FM&C)		
Scientific and Technical	OPNAV N091; MCCDC	ASN(RD&A)		
Test and Evaluation	OPNAV N091; MCOTEA	ASN(RD&A)		
Medical	OPNAV N093			XXXXXXXXXXXXXX
Reserve Affairs	OPNAV N095; HQMC M&RA	ASN(MR&A)		
Meteorology, Oceanography, GI&S	OPNAV N096	ASN(RD&A)		XXXXXXXXXXXXXX
Precise Time and Astrometry	OPNAV N096	ASN(RD&A)		XXXXXXXXXXXXXX
Religious Ministries	OPNAV N097			XXXXXXXXXXXXXX
Naval Nuclear Propulsion	OPNAV NOON			XXXXXXXXXXXXXX

APPENDIX J: Case Study 1: Employee/Member Self Service Integration

The development guide provides many options for integration. The case studies provide some real-world, practical examples of portal implementation. In this case, we will take a look at the Employee/Member Self Service (E/MSS) web site, a non-DON site, and show how it might integrate into the NEP.

Rewriting the entire web application is not an option. The DON does not own the site and it has to be accessible to other organizations, such as the Army and Air Force. In addition, the web application is designed for a full screen view as shown as follows and is not portal friendly.

Figure 30: E/MSS Opening Screen

Based on the above information, the main E/MSS login screen and application should appear full screen in a separate window, yet keep a small point of access available on the portal. The best choice of integration would be reference integration.

The options for reference integration are based on the discussion of portal content integration contained in the NEADG and are listed in the following table:

Table 37: E/MSS Reference Portlet Characteristics

Portlet Characteristic	Reference Integration Characteristics	E/MSS Portlet Characteristics
UFS Output	HTML or XML/XSL	HTML
Portal Look-and-Feel Integration	Recommended	The portlet is built for the portal and uses portal defined CSS. The application uses its own look and feel in a separate window.
Compatible with Portal Reverse Proxy (URL Rewrite)	Recommended	The portlet works correctly with the URL Rewrite enabled.
HTML BASE TAG for relative references vs. absolute references	Recommended	Used
Portal Rendering of XML/XSL to HTML (XSLT)	As an external service call only	Not applicable
Portal IFRAME compatibility	Recommended	The portlet is portal-iFrame compatible. The application is in a separate application-controlled browser window.
Mobile Code (Applets, ActiveX)	Allowed within TFWeb policies and guidelines	JavaScript is used in the portlet, so SSL is used.
Application Frames/Iframes	Supported	The application is in a separate, application-controlled browser window.
Popup child windows	Supported	The application is in a separate application-controlled browser window.
UNICODE Support	Recommended	Yes

Implementation

To implement Reference Integration, we need to design a UFS that displays a portlet that fits in the portal frame, behaves nicely, incorporates the portal look and feel, and provides a link to the main E/MSS app.

The reference integration template HTML file was downloaded from the OpSS and modified for this example.

Using the existing logo image of the E/MSS a simple reference integration UFS (in this a HTML file) was created. Typically this file should be hosted on the backend server and the file URL provided as part of the service metadata in the submission package. See the completed example below.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
    <title>
        Employee Member Self Service
    </title>
       <!--- The portal will automatically insert a portal CSS reference here if you
request INSERTSTYLE=Y when you submit your metadata package --->
    <!--- a base tag is required by the portal if relative URLs are used in this template
--->
       <base href="https://emss.dfas.mil/" target="_blank">
<script language="JavaScript" type="text/JavaScript"><!--
function TFW_openBrWindow(theURL,winName,features)
{
  window.open(theURL,winName,features);
}
//-->
```

```
</script>
</head>
<!---
instructions for using the reference portlet template:
1. update URLs.
2. modify text for application name and description.
3. replace reference to cnn.gif with your application logo.
4. application logo should be no bigger than 150 x 50 pixels in size.
5. application logo must have no more than 64 colors.
6. application logo must be in GIF or PNG format and have a transparent background.
(Refer to Photoshop help file for steps in Mapping colors to transparency). Note: Test
your logo appearance in the Dev Portal. Colors should be checked within the Dev Portal
with various templates.
7. change the title tag to reflect your application name

minimum requirements for a reference portlet:
1. all links must open in a new browser window.
2. base tag is required if relative URLs are used.
3. Your logo.
additional desirable requirements
1. incorporate the portal look and feel (InsertStyle=Y)
2. be compatible with the portal URL Rewriter (Rewrite URLs=Y)
--->
   <body leftmargin="0" topmargin="0" marginwidth="0" marginheight="0">
        <table width="400" border="0" cellspacing="5" cellpadding="0">
                <tr>
                        <td       width="8%"       align="center"       valign="middle"><img
src="images/mypay.gif"><br>
                        </td>
                        <td                     width="92%"                    align="right"
valign="bottom"><strong>Employee/Member Self Service </strong>(EMSS)
                        </td>
                </tr>
                <tr bgcolor="#999999">
                        <td colspan="2" style="height: 1px;">
                        </td>
                </tr>
                <tr>
                        <td height="26" colspan="2">
                                Welcome to Employee/Member Self Service. E/MSS allows you,
as a Department of Defense Military Member, Civilian Employee, Military Retiree or
Annuitant to make certain changes to your pay information.
                        </td>
                </tr>
                <tr>
                        <td height="24">       
                        </td>
                        <td align="right" valign="bottom">
                                <a                                                      href="#"
onClick="TFW_openBrWindow('https://emss.dfas.mil/mypay.asp','','')">
                                Access the EMSS Service
                                </a>
                        </td>
                </tr>
        </table>
   </body>
</html>
```

This code displays the following resizable portlet.

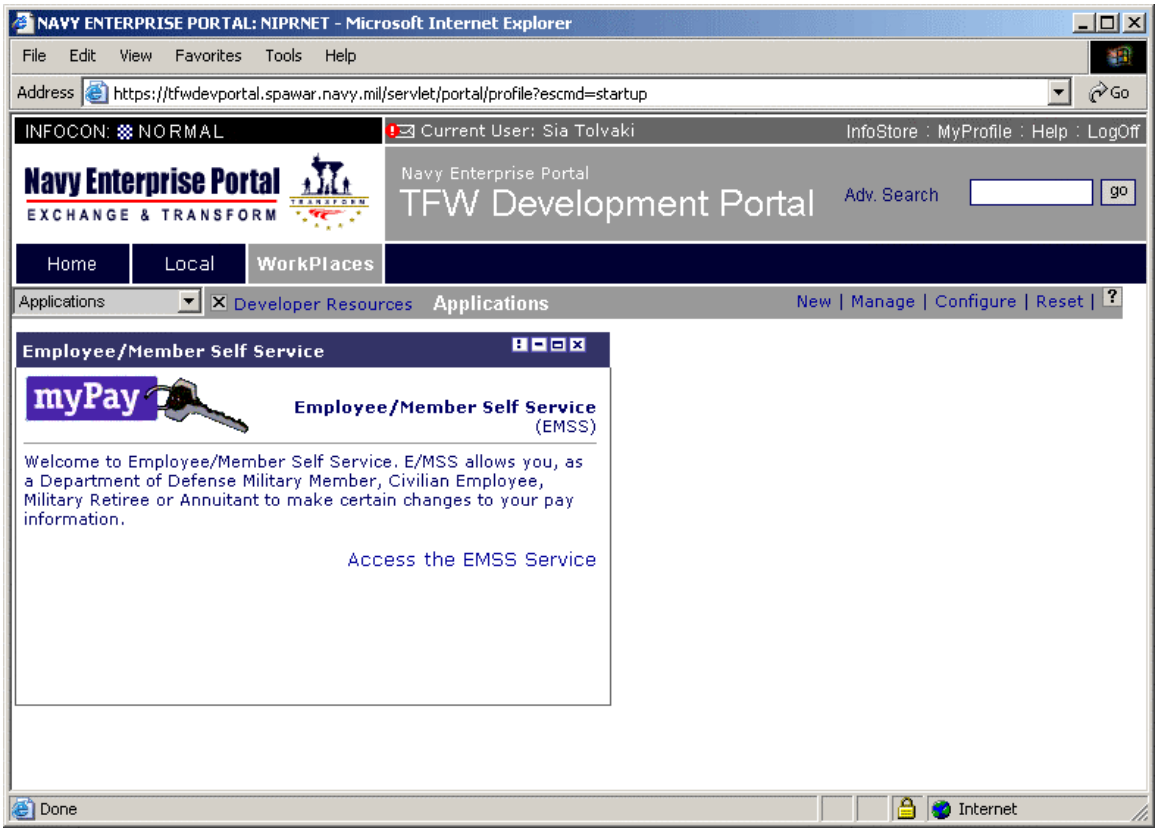


Figure 30: Portal Friendly Reference Integration Portlet

The portlet integrates well within the portal. Particular attention was paid to the logo image to reduce the file size and to give it a transparent background that will allow the best integration with the various portal templates.

The portlet is also completely compatible with the Portal URL Rewriter. In this case all of the URLs in the HTML file will be rewritten except for the javascript that was used to launch the full screen application. The javascript URL is hidden from the rewriter and thus when opened, the EMSS application runs completely outside of the portal.

The portlet takes advantage of another portal feature known as InsertStyle. When this feature is enabled, a reference to the portal Cascading Style Sheet (CSS) is automatically inserted into the portlet HTML. This causes the background colors and font size and color to match the rest of the portal.

Another technique that may be used to accomplish the CSS insertion is to request PRI data as part of the service metadata. Your application must decode the PRI ClientStyle element and insert it into the HTML output. For simple reference integration, the InsertStyle feature eliminates the need to receive and decode the PRI data.

The reference portlet displays the following when the portal user selects the Command Center template (dark background, with low contrast text colors):

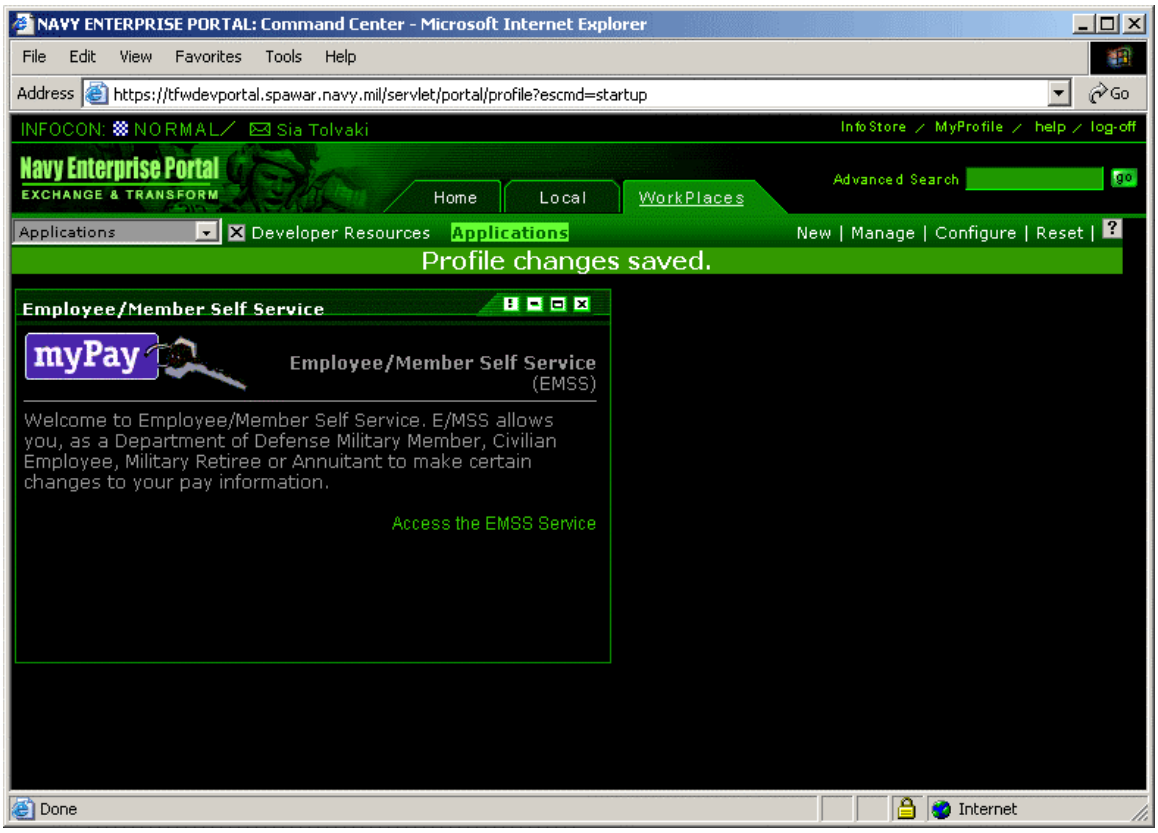


Figure 31: Reference Integration Portlet Utilizing Portal CSS

Now, as the user selects a different template, a different CSS is used. When the user clicks on "Access the EMSS Service" a separate window pops up with the main E/MSS application. The portlet adheres to the user-selected color and font schemes as intended; however, the main application does not. Although this is a simple case, it demonstrates a few portlet programming needs, such as adherence to portal compatibility and use of the portal-defined CSS.

APPENDIX K: Case study 2: NTIRA Web Project Case Study (Phase I)

NTIRA POC: Fisher, Cabell C. - lead developer/architect for NTIRA - fishercc@spawar.navy.mil

Business Analysis

The Naval Tool for Interoperability and Risk Assessment (NTIRA) has emerged as a high profile and valuable Navy software application. NTIRA exists as a suite of back office applications that provide access to a capabilities-based view of Battle Groups, Platforms (Ships) and their supporting systems, and ashore infrastructures. The NTIRA application domain targets the DoN resource allocation and prioritization business community. Functionally, NTIRA is designed to assist planners and decision makers through rapid, semi-automated assessment of budgetary realignments and constraints and their impact on end-to-end war fighting mission capabilities. The goal of the first phase NTIRA Web Project was to migrate high value views and functions from an existing desktop/client-server application to the NEP. The timeline for implementing the first phase was one month from analysis to delivery. The following UFSs were implemented in this timeframe and exposed to the NEPl: Fiscal Reporter, Battle Group Mission Status, Mission Status by Revision, Mission Status by Platform, and Configuration Reporter.

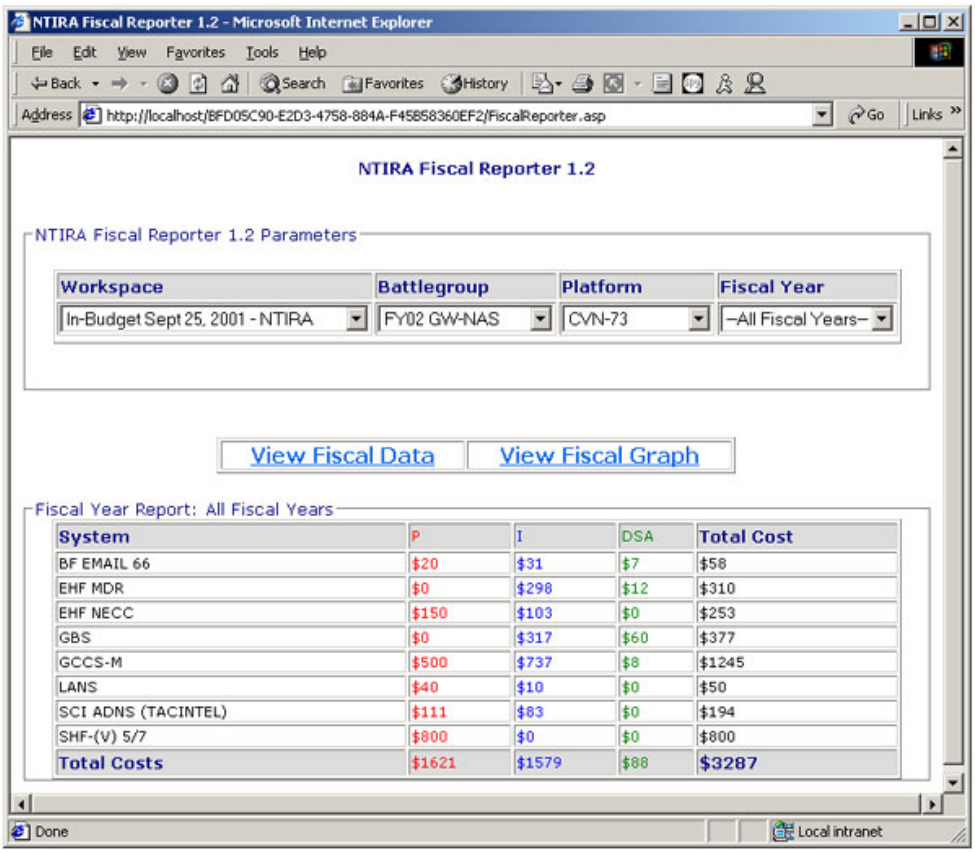


Figure 32: Fiscal Reporter UFS – Fiscal Data View

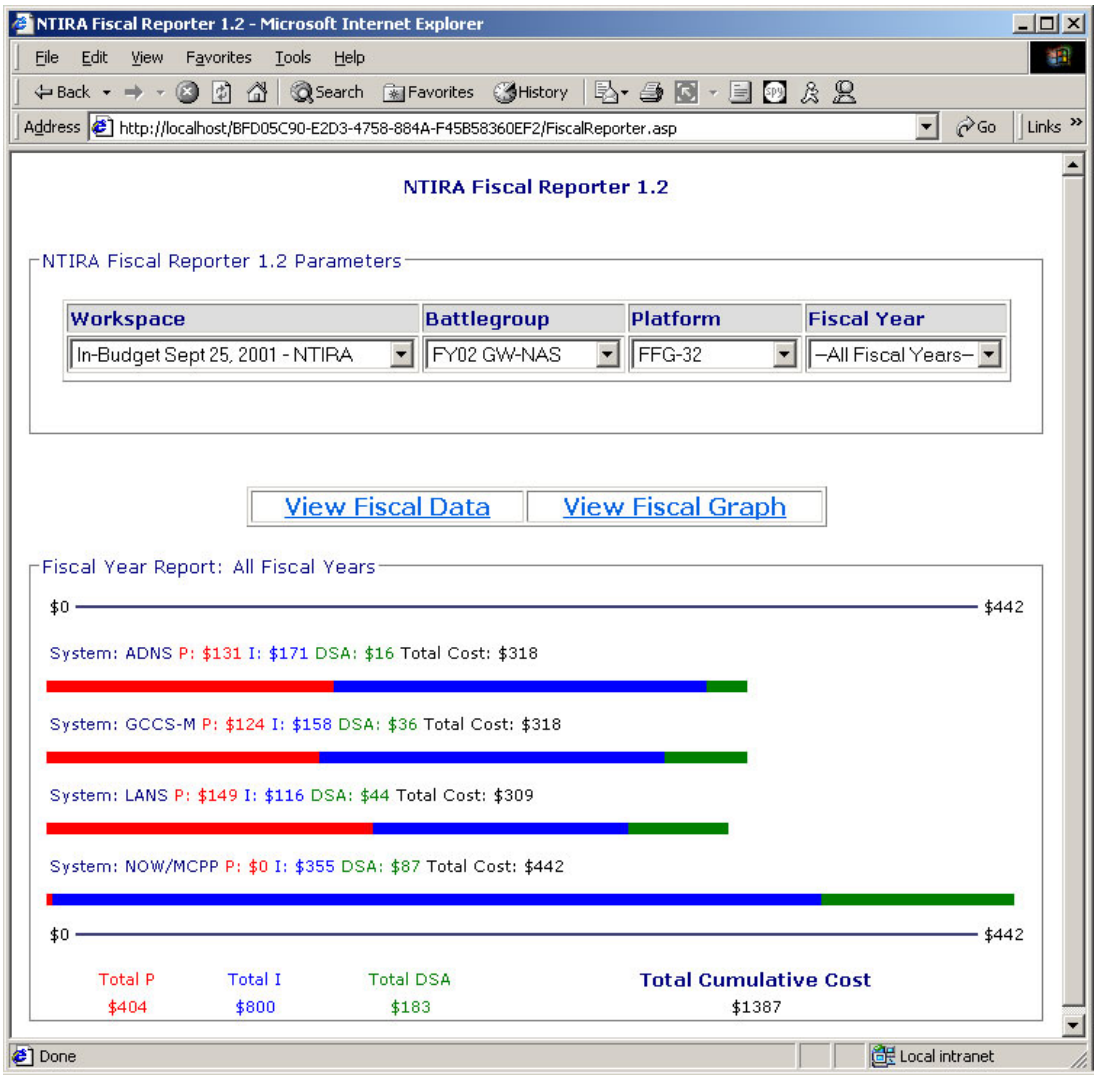


Figure 33: Fiscal Reporter UFS – Fiscal Graph View

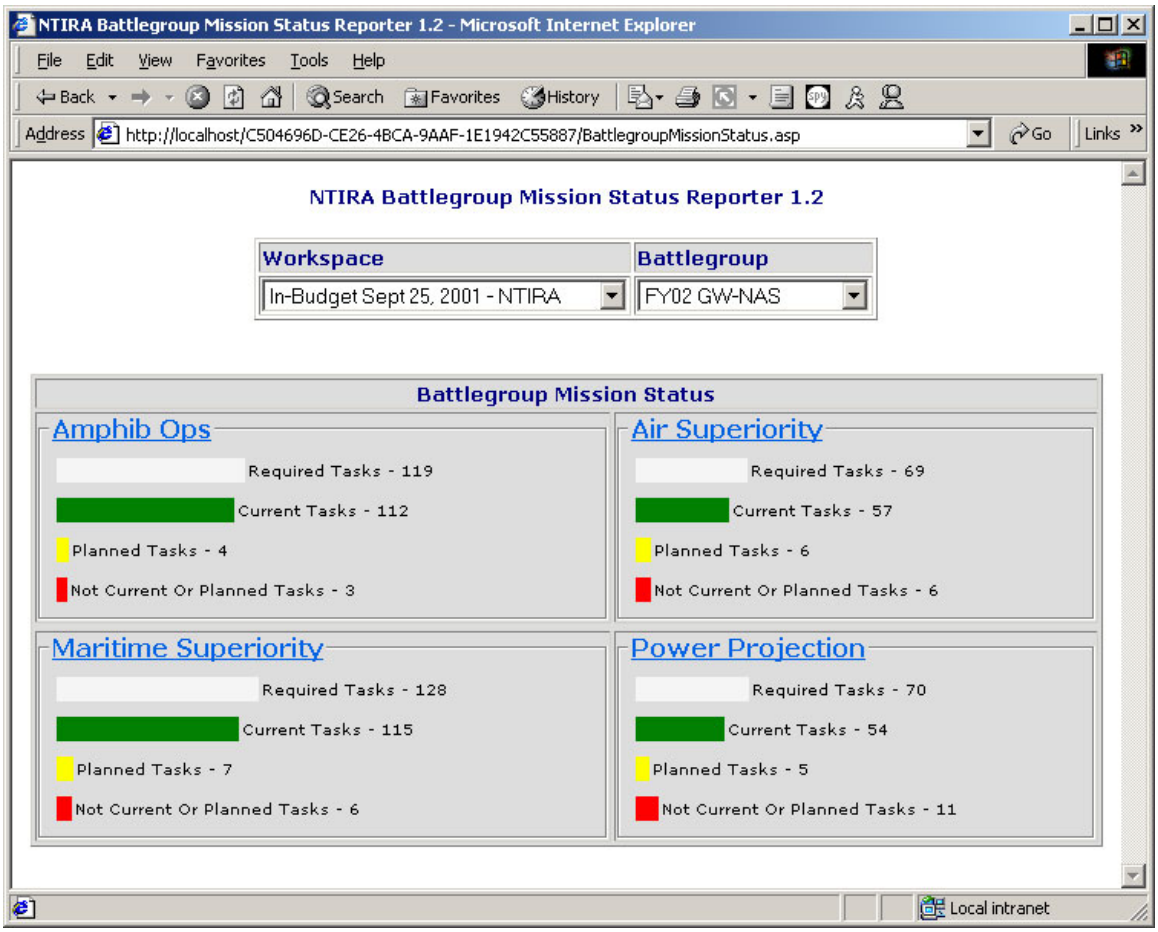


Figure 34: Battle Group Mission Status UFS

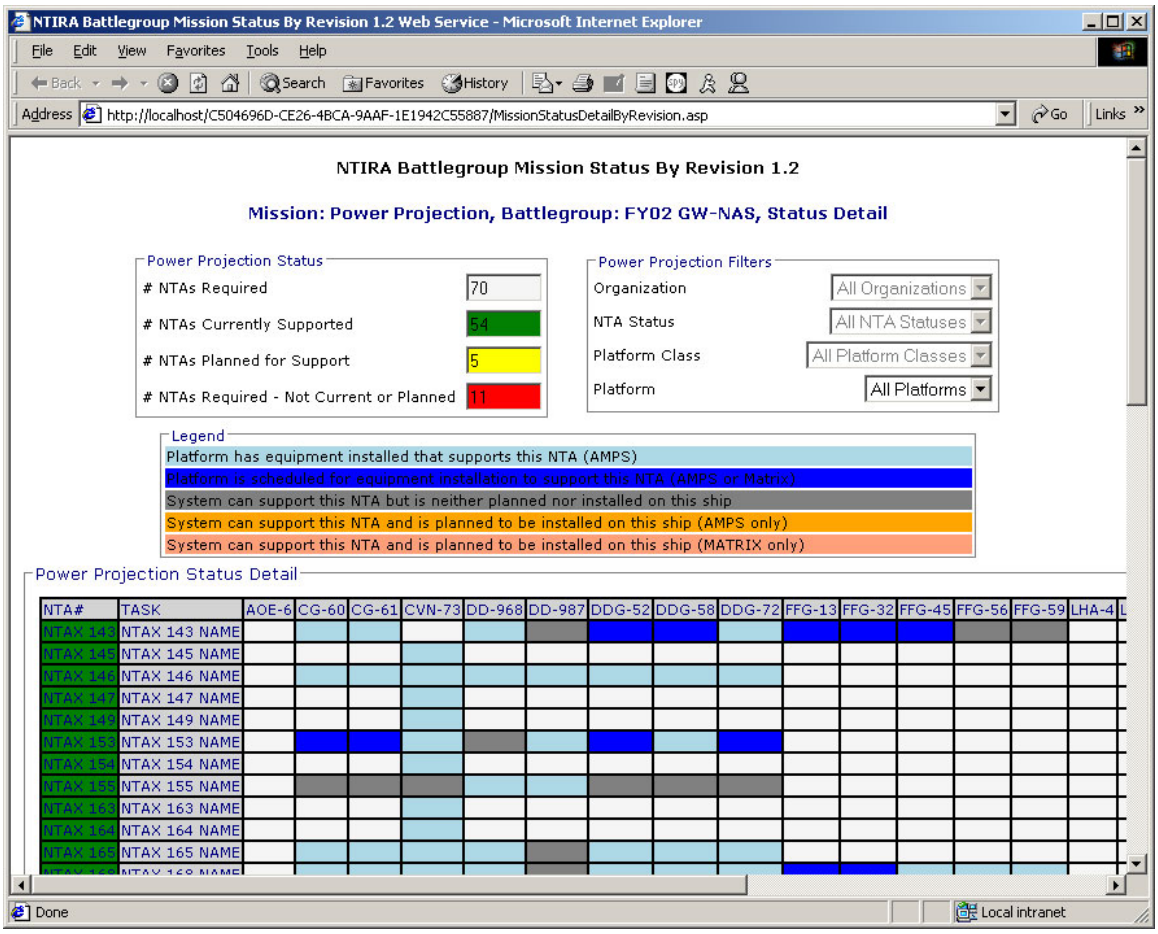


Figure 35: Mission Status by Revision UFS

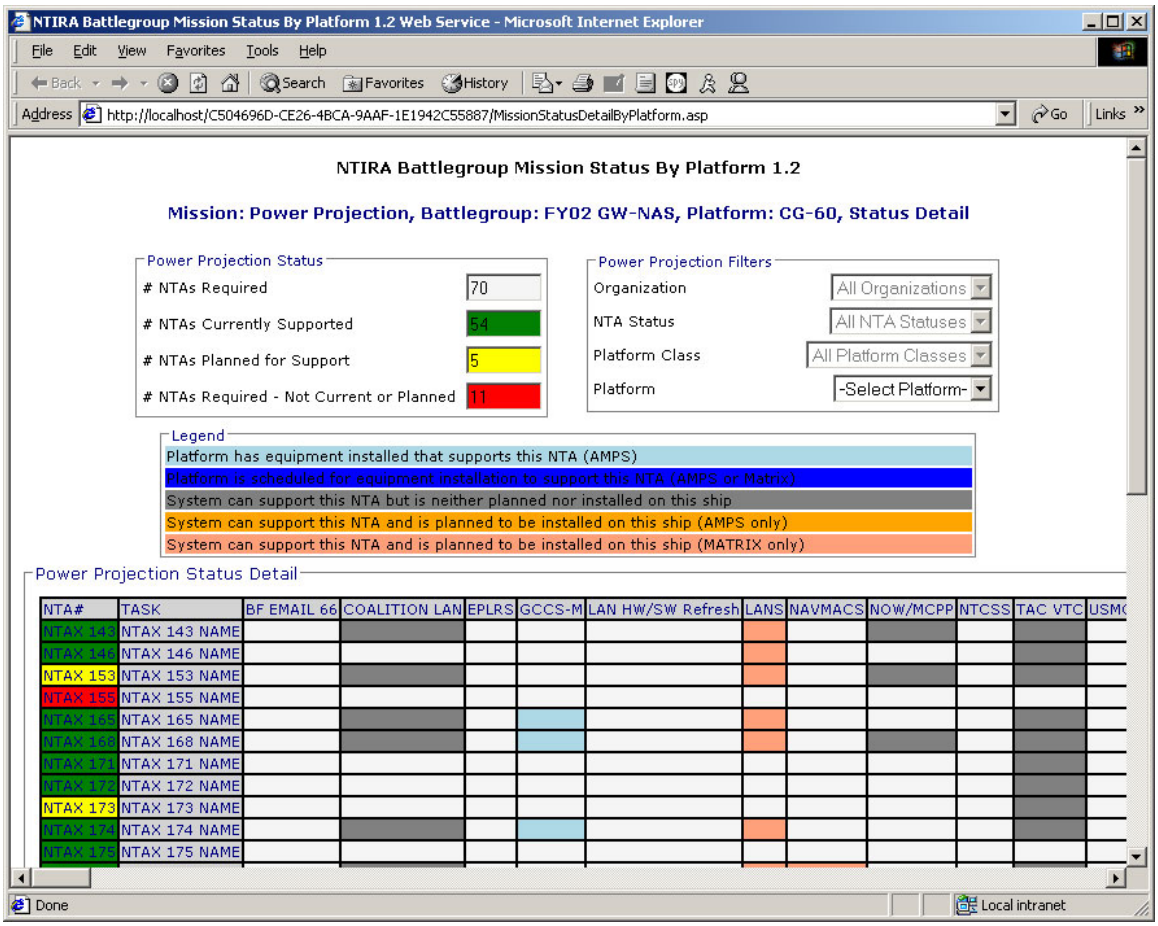


Figure 36: Mission Status by Platform UFS

Each of the UFSs interacts with SOAP-based data-oriented services that expose application business logic and data to the UFS.

Technical Analysis

Existing "Fat Client" Software Architecture

The existing desktop/client-server NTIRA implementation provides a set of Executive and Analyst functionalities accessible through a Visual Basic client interface. Both the software that processes the client interface and the SQL Server database resides on the client machine and is referred to as the "fat client." The existing desktop/client-server implementation limits scalability, forces complex data replication, and requires a client installation for each version release.

Migrating to the Navy Enterprise Portal

The goals of the NTIRA web project were to:

- Re-architect the existing desktop/client-server implementation and migrate to a defined set of methods and business objects.

- Expose the methods (business logic and data) as logically defined SOAP-based web services.
- Update the existing desktop/client-server implementation to communicate via SOAP to the NTIRA application server to process live updates.
- Create UFSs that provide web-based views of the existing Executive and Analyst views available in the desktop/client-server implementation. Expose these UFSs to the NEP.

Implementation

NTIRA Web Software Architecture

The NTIRA Web Project will be implemented in phases, the first of which involves exposing the core application business logic and data as SOAP-based web services. This web service architecture allows us to build on existing encapsulated business logic and make the NTIRA data available and presentable to the NEP. The NTIRA web application can be broken into three layers: data, business logic, and presentation.

Data Layer

The data layer was implemented as a SQL Server 2000 database. For the initial phase, we used the same database schema that powered the existing desktop/client-server implementation. Fortunately, the majority of the NTIRA application's business logic and data retrieval was already encapsulated in stored procedures. To build on this foundation will quickly make existing functionality accessible over the web.

Business Logic Layer

The business logic layer was implemented as a set of Java session beans corresponding to the functional NTIRA modules (Fiscal, Capability, Configuration). The initial phase concentrated first on simply wrapping the existing stored procedures and exposing them as methods of the session beans. Additional business logic was added to the methods to accommodate situations where business logic was coded into the VB Forms of the desktop/client-server implementation. A library was created of generic functions including one that would generically transform a Resolve Set into an XML document. The session beans were installed in BEA Web Logic 6.1 Java Application Server. The BEA Web Logic 6.1 Application Server allows the methods of session beans to be exposed as web services invokable via SOAP. Use of a freeware tool called ANT (see http://jakarta.apache.org/ant/) enabled configuration of the deployment descriptor files for the session beans and had BEA automatically generate the WSDL documents and expose the methods as web services. Capitalizing on the built-in features of BEA Web Logic server would quickly expose the NTIRA business logic as DOSs.

Presentation Layer

The display layer for the first phase was implemented as a set of UFSs. The UFSs were implemented as Active Server Page (ASP) and Java Server Page (JSP) components. The ASP and JSP components communicate with the DOSs via SOAP. The UFSs build SOAP client messages and send them to the NTIRA business logic layer. The session bean methods are exposed as SOAP Servers, so our UFSs can execute them, consume the returned XML data, and return a dynamic HTML interface to the portal. In addition to the ASP and JSP UFSs, several visual web services were exposed to the NEP. These visual web services were implemented as SOAP-based web services that returned XML data and XSLT style sheets.

NTIRA Web Components

The following diagram depicts the software components of the NTIRA Web Project and how they fit into the NEP environment. The areas in gray represent physical software components and development tasks associated with the project.

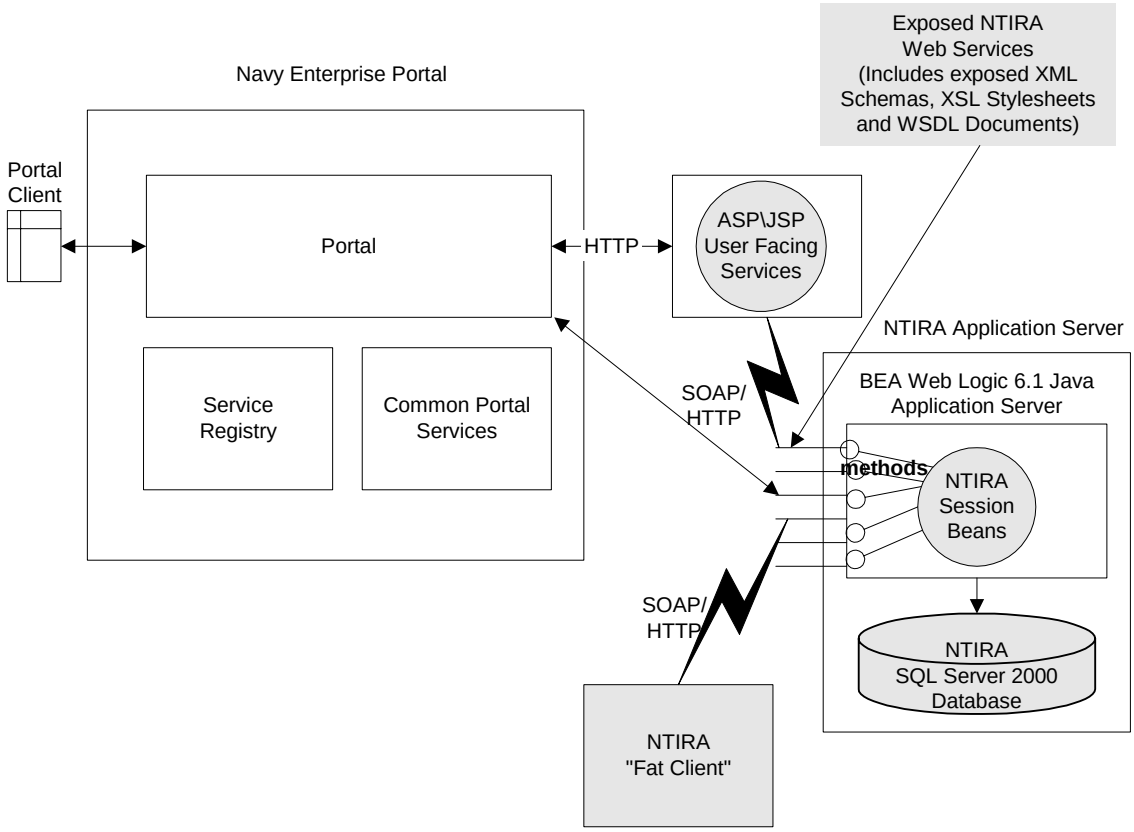


Figure 37: NTIRA Software Components

Summary

The initial phase of the NTIRA web project took exactly one month. Three members of the TFWeb team worked full-time with key members of the NTIRA development team to provide analysis, support, and development efforts. Five high-value NTIRA views were moved to the web and exposed to the NEP and over thirty data-oriented services were made available over the web accessible via SOAP.

APPENDIX L: NMCI Application Rule Set

Justification IDs for Rejected and Failed Material (See http://cno-n6.hq.navy.mil/navcio/file/NMCI-RuleSet.doc

The following codes explain why submitted material has a final status of Rejected.

ID #	Description	Status
1. Win2K Incompatible	The software is not compatible with the Windows 2000 Operating system. This means that the candidate software either will not run properly under Windows 2000 or that it interferes with the normal functionality of the operating system.	FAIL
2. Group Policy Object Violation	The application is not compatible with the GPO security rules for the Gold Disk and a solution is not viable. For instance, if the candidate application requires full control of the c:\winnt folder in order to run, this violates NMCI enterprise policy governing connection to the NMCI network, thus disqualifying the application.	FAIL
3. No duplication of standard seat services	An application installed on all the new computers obsoletes the requested one (Example: Word replaces WordPerfect).	Rejected (NRFC)
4. DoNFirewall Violation	The joint ISF and government Information Assurance Tiger Team has recommended the application as a Category 9 legacy applications/system, it is non-compliant with Navy/Marine Corps firewall policy, and it will never migrate behind Boundary 1 (i.e., into the trusted NMCI enclave). This means that the candidate application, including third-party stand-alone software, is disqualified for violating Navy or Marine Corps network security policy. Such an application may be a candidate for major re-engineering or retirement.	Rejected (NRFC)
5. N/A	No longer used.	
6. Games	The candidate application is a "game," as defined in the Application Rulebook by the ISF, PEO-IT, and the PMO. NMCI and the claimant have agreed not to test recreational game software for certification.	Rejected (NRFC)
7. Freeware/ Shareware	The candidate application is "freeware" or "shareware", as defined in the Application Rulebook by the ISF, PEO-IT, and the PMO. NMCI and the claimant have agreed not to test freeware or shareware for certification.	Rejected (NRFC)
8. Beta/Test	The candidate application is a "beta" or a "test" version, as defined in the Application Rulebook by the ISF, PEO-IT, and the PMO. NMCI and the claimant have agreed not to test pre-release versions of software for certification.	Rejected (NRFC)
9. Application Development Software	The candidate application is "application development" software, as defined in the Application Rulebook by the ISF, PEO-IT, and the PMO. If the site is a Research, Development, and Testing (RD&T) site, the candidate application can be ordered for a developer seat. Otherwise, NMCI and the claimant have agreed not to test application development software for certification.	Rejected (NRFC)

ID #	Description	Status
10. Agent Software	The candidate application is "agent" software, as defined in the Application Rulebook by the ISF, PEO-IT, and the PMO. Certain ActiveX controls (and others that automate Windows processes) may not interact correctly with the local firewall system or may pose an information security risk.	Rejected (NRFC)
11. Gold Disk Incompatible	The application software is not compatible with the standard "Gold Disk" software, excluding the Windows 2000 operating system. This means that the candidate application does not interact properly with one or more of the set of applications that have been selected to be installed on all of the claimant's new PCs.	FAIL
12. Requires Internal Peripheral Hardware	The submitted application requires hardware that must be installed inside the computer case, such as a PCI, ISA, or AGP card. A new component may be connected only to an external terminal (serial, USB, parallel port, etc.)	FAIL
13. Personal, non-mission, or non-business related software	The candidate application is "personal, non-mission, or non-business related" and is therefore prohibited on the NMCI environment.	Rejected (NRFC)
14. 8 or 16-bit applications	The DoN has determined that 8- and 16-bit applications will not migrate to NMCI.	Rejected (NRFC)

APPENDIX M: NEP Developer's Integration Checklist

☐ Establish contact with appropriate Echelon Level AMCS & AMTS POC.

☐ Utilize the NEP Development portal systems as required. See section 3.1.1 NEP Service Certification Process.

☐ FAM and AMCS rationalization processes have been followed. See section 1.1.3 Functional Area Manager Rationalization and section 3.1.5 NEP Service Rationalization for more information.

☐ XML usage compliant with the DON policy on the use of XML and the DON XML Developers Guide (http://quickplace.hq.navy.mil/navyxml).

☐ XML schemas are registered with the DoD XML Registry (http://xml.dod.mil/xmlreg/user/index.cfm)

☐ PKI certificate is in the hierarchy based on the DoD Class 3 Root CA certificate. See APPENDIX G: Application Security for more information.

☐ IATO or ATO obtained from the appropriate DAA for the software developer. Application and database owners creating services for the NEP must ensure their application/databases comply with all Navy and DoD directives for networked systems. Access through the NEP does not preclude or negate responsibility for compliance with Information Assurance, 508, and other all other applicable directives.

☐ Application registered in the DON Applications & Database Management System (DADMS) (https://www.dadms.navy.mil).

☐ Enter Application in ISF Tools Database and receive NMCI Request for Service number for NIPRNET application (See 3.2 NMCI Integration Process and ISF).

☐ Submit application/service migration package via the "Submission Package" portlet on OpSS. Includes the following items.

☐ Service Registry Metadata (See 3.1.7.1 Service Registry Metadata).

☐ Test plan and cases.

☐ Provide a temporary testing user login with access to non-administrator portions of the application.

☐ Portlet Migration Plan to achieve content integration (or justification why that is not required) with appropriate milestones.

☐ Provide mobile code controls. Required only if the mobile code controls need to be hosted on NEP (mobile code hosting is optional). See APPENDIX N: Navy Mobile Code Policy, and section 3.2 NMCI Integration Process and ISF for additional guidance.

☐ Provide production service instance deployment information. See section 3.1.11 NEP Service Deployment for more information.

APPENDIX N: Navy Mobile Code Policy

Section 5 of CNO N614 / HQMC C4 NAVY- MARINE CORPS UNCLASSIFIED TRUSTED NETWORK PROTECTION (UTN*Protect*) POLICY Version 1.0, dated 31 October 2002

Mobile Code Policy

5.1 Implementation: Only mobile code that conforms to the definitions and conditions specified in this section is authorized for transit from an untrusted network to a trusted network across the trusted network boundary.

> NOTE: The intention of this policy is to force 'mobile code' to be treated the same as any other executable code. This means equivalent, but not more restrictive conditions will be imposed on the use of mobile code than are required for other forms of executable code. A user makes a conscious decision when executing an .exe or .bat: The same "conscious decision" must be made when executing mobile code. Execution is not the concern - 'Automatic' execution is the concern.

5.1.1 Permissions contained in Section 5.4 will not be set by default: There must be a valid operational requirement, within each trusted network, that requires their use. A trusted network DAA is responsible for determining the operational requirements for that network.

5.2 Background: Mobile code is software obtained from remote systems outside the trusted network boundary, transferred across a network, and then downloaded and executed on a local system without explicit installation or execution by the recipient. A trusted network is defined as a network under the control of a single organization with the responsibility to define and implement security controls. In this regard, mobile code optimizes functionality while minimizing bandwidth. For example, it "uses" local workstation resources to animate graphics, to enable word processing or perform math calculations without having to transfer large amounts of data across the network. There are many powerful reasons to use mobile code and its use in Navy Networks is generally consistent with its use across DoD. Unfortunately, it can also be used maliciously to disrupt or degrade operations, so measures to secure and control mobile code technologies are in the best interests of the Navy and Marine Corps.

5.3 Navy-Marine Corps Policy Regarding Mobile Code: The intent of the evolving DoD and Navy-Marine Corps Mobile Code Policy is to afford the use of all categories of mobile code in appropriate situations with suitable mitigation techniques (e.g. public key infrastructure, trusted source, assured path). This Navy-Marine Corps Mobile Code Policy will be updated to reference and implement formal DoD policy directives when promulgated.

5.4 Authorized Use of Mobile Code:

> Note 1: If procedures are in place where "mobile code software" (ActiveX, Java Applets, or other 'executables') can only be installed/executed with the explicit acknowledgement and understanding of the user, such usage falls outside the scope of this policy. As a representative implementation, if a system administrator ensures that all the 'browsers' in his/her trusted network are configured to prompt the user and require an 'execute' affirmation, then the use of ActiveX or Java Applets are not subject to the conditions prescribed in this section.
>
> Note 2: ALL caveats listed in the 'Required Conditions' column must be met prior to use of code originating outside the trusted network boundary.

Technology	Required Conditions
ActiveX	1. Allowed only over assured channels 2. Allowed only from trusted sources 3. Code must be signed with a DOD PKI Code signing certificate (Available pending release V3.1 of DoD PKI)
Java	1. Allowed only over assured channels 2. Allowed only from trusted sources
JavaScript	Allowed (No Required Conditions)

5.5 Definitions: The following standard definitions are applicable to this section:

5.5.1 Assured Channel: A network communication link that is protected by a security protocol providing authentication and data integrity. The following protocols and mechanisms are sufficient to meet the requirements of authentication and data integrity protection for an assured channel: Internet Protocol Security (IPSec), Secure Socket Layer (SSL), Transport Layer Security (TLS), Secure Multipurpose Internet Mail Extension (S/MIME), or digital code signing using a DoD approved PKI code signing certificate.

5.5.2 Mobile Code: Software obtained from remote systems outside the trusted network boundary, transferred across a network, and then downloaded and executed on a local system without explicit installation or execution by the recipient.

5.5.3 Trusted Source: A source that is adjudged to provide reliable software code or information and whose identity can be verified by authentication. The following mechanisms are sufficient to validate the identity of a trusted source: connection via digital signature over the mobile code itself using a DoD-approved PKI code signing certificate, or authentication of the source of the transfer by DoD public key certificate (e.g., S/MIME, SSL server certificate from an SSL web server).

5.6 Points of Contact:

CNO N614

Mr. Terry Danner
CNO Staff N614212
Presidential Tower 1 Suite 5630
2511 South Jefferson Davis Hwy
Arlington, VA 22202-3926
(703) 601-1491 / DSN 329
NIPRNET: danner.terry@hq.navy.mil
SIPRNET: danner.terry@cno.navy.smil.mil

SPAWAR PMW161:

Mr. Ed Burr
COMSPAWARSYSCOM (PMW161-3B)
4301 Pacific Hwy

San Diego, CA 92110-3127
(619) 524-7519
NIPRNET: edgar.burr @navy.mil
SIPRNET: burre@spawar.navy.smil.mil

Ms. Edie Stearns
COMSPAWARSYSCOM (PMW 161-3B1)
4301 Pacific Hwy
San Diego, CA 92110-3127
(619) 524-7824
NIPRNET: edie.stearns@navy.mil
SIPRNET: stearns@spawar.navy.smil.mil

MITNOC:

MAJ Glen Sharlun
SPCA, MITNOC
(703) 784-5300/DSN 278-3698
Fax: (703) 784-3477/DSN 278-3477
NIPRNET: sharlungf@noc.usmc.mil
SIPRNET: sharlungf@noc.usmc.smil.mil

Ms. Bonnie J. Bienz
SPCA, MITNOC
(703) 784-3698/DSN 278-3698
Fax: (703) 784-3477/DSN 278-3477
NIPRNET: bienzbj@noc.usmc.mil

SIPRNET: bienzbj@noc.usmc.smil.mil