



# FAA Telecommunications Infrastructure NEMS User Guide

20 November 2013

CDRL: F034  
CAGE Code 91417  
Document Control Number: 7042480  
Contract Number: DTFA01-02-D-03006



**Prepared for:**  
Federal Aviation Administration  
800 Independence Avenue, SW  
Washington, DC 20591

**Prepared by:**  
**HARRIS CORPORATION**  
Government Communications Systems  
PO Box 37  
Melbourne, FL USA 32902-3700



## REVISION RECORDS

REVISION	DATE	DESCRIPTION
—	11 July 2013	Initial Release.
A	10 October 2013	Incorporation of FAA comments on initial release.
B	20 November 2013	Incorporation of FAA comments

## TABLE OF CONTENTS

<u>Paragraph</u>	<u>Title</u>	<u>Page</u>
1	SCOPE.....	1
1.1	Summary.....	1
1.2	Background.....	2
2	REFERENCES.....	3
2.1	Government Documents.....	3
3	NEMS OVERVIEW.....	4
4	INTERFACE DESIGN CHARACTERISTICS.....	7
4.1	General Characteristics.....	7
4.2	Network Interface Characteristics.....	8
4.3	Software Interface Characteristics.....	9
4.3.1	Messaging Service.....	10
4.3.1.1	Messaging Usage Guidance.....	13
4.3.2	Service Level Security Service.....	15
4.3.2.1	Service Level Security Usage Guidance.....	15
4.3.3	Mediation Service.....	15
4.3.3.1	Mediation Usage Guidance.....	17
4.3.4	Subscription Catalog Service.....	17
4.3.4.1	Subscription Catalog Usage Guidance.....	17
5	NEMS ADMINISTRATION PROCESS.....	18
5.1	Network Access Steps.....	18
5.2	NEMS Access Steps.....	18
5.2.1	Publish/Subscribe Consumer Steps.....	18
5.2.2	Request/Response Consumer Steps.....	19
5.2.3	Publish/Subscribe Producer Steps.....	19
5.2.4	Request/Response Producer Steps.....	20
5.2.5	Producer Disconnect.....	20
5.2.6	Consumer Disconnect.....	20
5.3	NEMS Navigator Access Instructions.....	21
6	NEMS DEVELOPMENTAL SUPPORT.....	21
7	NEMS OPERATIONAL SUPPORT.....	21
8	NOTES.....	23
8.1	Abbreviations and Acronyms.....	23

## LIST OF APPENDICES

<u>Appendix</u>	<u>Title</u>	<u>Page</u>
A	NEMS JUMPSTART KIT PRODUCER AND CONSUMER OVERVIEW .....	A-1

## LIST OF ILLUSTRATIONS

<u>Figure</u>	<u>Title</u>	<u>Page</u>
1.2-1	Net-Centric Data Exchange .....	3
3-1	NEMS Logical Architecture View .....	5
3-2	NEMS Deployment Architecture View .....	6
3-3	NEMS Current Deployment Status .....	7
4.2-1	FTI-NEMS Network Interface .....	8
4.3-1	Service Access Points (SAPs) for NEMS Services.....	10
4.3-2	NEMS Service Categories .....	10
4.3.1-1	Static Subscriptions Messaging .....	11
4.3.1-2	Dynamic Subscriptions Messaging .....	12
4.3.1-3	Web Service Messaging .....	13
7-1	Operational Support Organization .....	22

## LIST OF TABLES

<u>Table</u>	<u>Title</u>	<u>Page</u>
2.1-1	Government Documents.....	3
4.2-1	FTI-NEMS Network Interface Protocols.....	9
8.1-1	Acronyms and Definitions .....	23

## 1 SCOPE

This document provides technical information for system engineers and software developers of National Airspace System (NAS) Enterprise Messaging Services (NEMS) producers and consumers, to facilitate an initial understanding of the technical details regarding the access and utilization of the capabilities and services offered by the NEMS, as well as the NEMS administration processes for obtaining access to NEMS run-time services. This information covers currently deployed and soon to be deployed capabilities. Additional planned capabilities are expected to be available over the next 1-2 years and this document will be updated to include these capabilities as they near deployment.

This document is intended to be used primarily by software development organizations and is intended to supplement other System Wide Information Management (SWIM)/NEMS documentation being developed. The other documentation provides high level NEMS capability descriptions, architecture usage patterns, functional and performance capabilities, interface descriptions, and ordering information.

This document does not provide information about actual content that is provided via NEMS messaging services. Information about content is provided via service description documents (Web Service and JMS as applicable) which are located in the NAS Service Registry/Repository (NSRR).

The following existing SWIM/NEMS documentation should be referenced for additional information regarding NEMS capabilities and services.

SWIM Solutions Guide - The purpose of this document is to provide Federal Aviation Administration (FAA) program managers and architects with an understanding of how the SWIM Enterprise Service Bus (ESB) provided by NEMS can be used to meet program needs. The document is also intended to provide material that program managers can provide to contractors who will be bidding to build NAS systems that will need to use NEMS.

### 1.1 Summary

The NEMS provides net-centric messaging services that enable information exchange and Enterprise Services sharing among NAS applications in support of NextGen initiatives. This document contains the following sections:

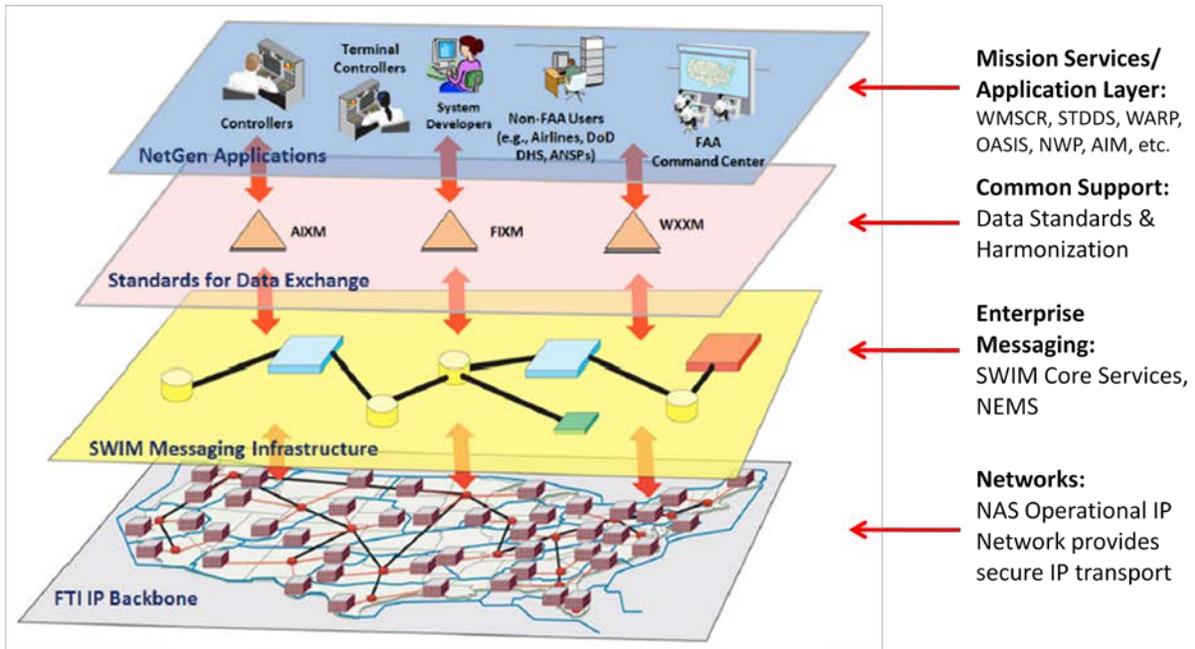
- Section 1, SCOPE, describes the intent of this document and introduces NEMS.
- Section 2, REFERENCES, lists applicable reference documents.
- Section 3, NEMS OVERVIEW, provides an overview of the NEMS infrastructure.
- Section 4, INTERFACE DESIGN CHARACTERISTICS, documents the interface design characteristics, such as points of demarcation, API usage, applicable supported standards, and supported protocols.

- Section 5, NEMS SYSTEM ADMINISTRATION PROCESS, describes procedures for obtaining access and connecting to NEMS services.
- Section 6, NEMS DEVELOPMENTAL SUPPORT, provides information regarding Harris support of producers and consumers during the on-ramping development phase.
- Section 7, NEMS OPERATIONAL SUPPORT, provides information related to customer support for NEMS services during the operations phase.
- Section 8, NOTES, abbreviations and acronyms used in this document.
- Appendix A provides information related to the Data-Exchange Jumpstart Kit (DJK).

## **1.2 Background**

At the heart of the NextGen concept is the information-sharing component which is being developed by the SWIM program. NEMS provides some of the core services under the SWIM program including messaging, service management, service-level security, and mediation services. NEMS provides a scalable system allowing NextGen to adapt to growth in operations as well as shifts in demand. It enables a publish/subscribe model, as well as a request/response model, serving as the foundation for robust, efficient, secure, and timely transport of information to and from a broad community of users and individual subscribers.

A net-centric enterprise tier approach enables NextGen applications to securely exchange data, in standards-based formats, using the SWIM NEMS messaging layer (layer 7) infrastructure that communicates via the FAA Telecommunications Infrastructure (FTI) NAS Ops IP Network (layer 3) Internet Protocol (IP) services as shown in Figure 1.2-1. Producers and consumers are “on-ramped” (i.e. connected) to the NEMS platform in order to exchange content via the NEMS. NEMS messaging and producer to consumer data flows are controlled and monitored from the FTI Network Operations Control Center (NOCC) utilizing NEMS service management components. NEMS security controls are managed from the FTI Security Operations Control Center (SOCC).



**Figure 1.2-1. Net-Centric Data Exchange**

The NEMS is currently operationally supporting various producers and consumers internal and external to the NAS. Additional producers and consumers are currently in the process of on-ramping to the NEMS. Additional NEMS capabilities identified by the SWIM program are currently being ordered and developed and will be deployed over the next five years to support NextGen initiatives.

## 2 REFERENCES

The following documents constitute a part of this document to the extent specifically set forth herein.

### 2.1 Government Documents

The Government documents listed in Table 2.1-1 a part of this document to the extent specified herein.

**Table 2.1-1. Table Title Here (Initial Caps)**

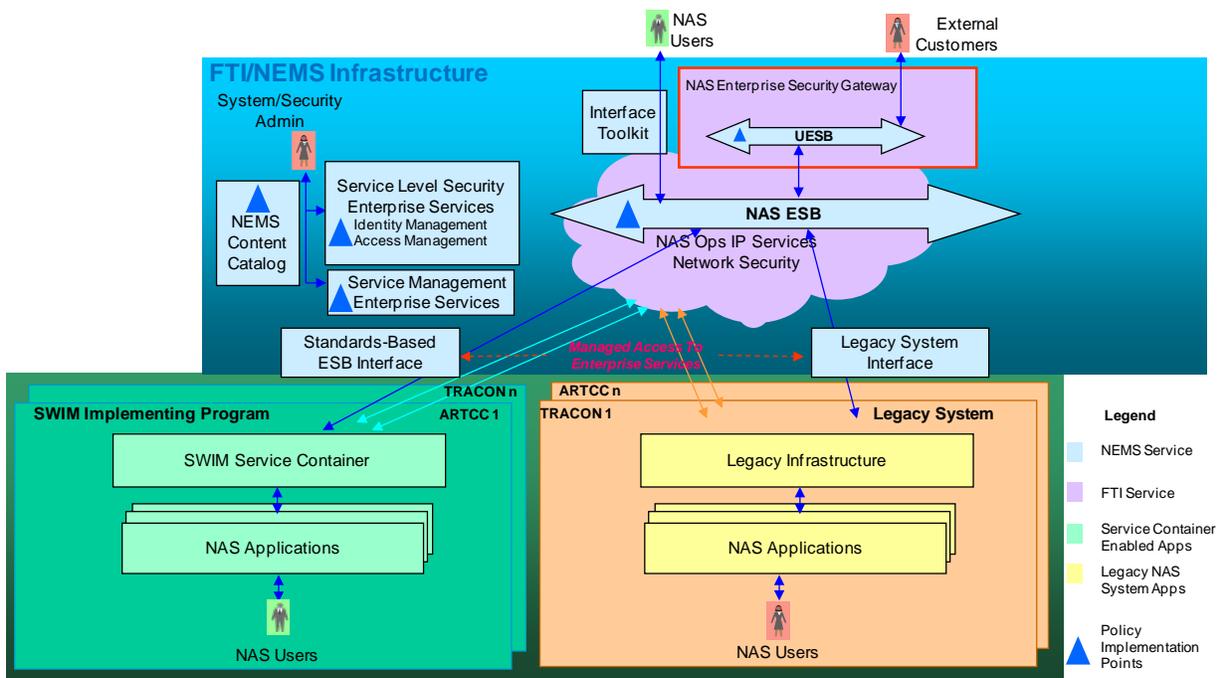
IDENTIFIER	TITLE	DATE	POC
Revision 2D	FTI Operational Network IP User Guide	May 2010	FAA FTI Program Office
Revision 1.0	SWIM Solution Guide for Segment 2A	June 2013	SWIM Program Office

Revision 3	FTI NAS Boundary Protection System (NBPS) User's Guide - Volume II For Non-NAS Users	July 2012	FAA FTI Program Office
------------	---	-----------	---------------------------

### **3 NEMS OVERVIEW**

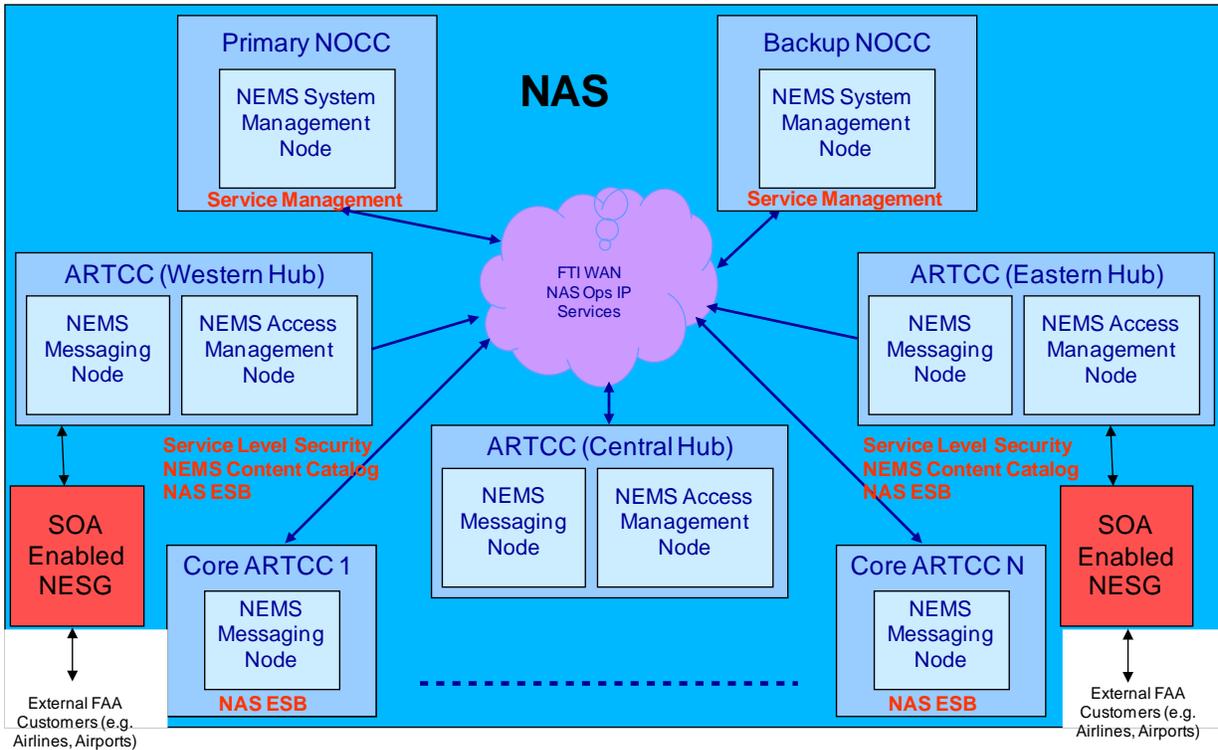
The NEMS system, utilizing a net-centric architecture shown in Figure 3-1, provides the NAS ESB that fosters the exchange of information, via enterprise services, among producers and consumers throughout the NAS and external to the NAS via the NESG. The NEMS establishes an FAA enterprise-level data exchange service that enables producers and consumers to exchange non-mission critical content and share services, with minimal impact to existing NAS applications. Exchange of mission critical information among mission critical applications is accomplished via private messaging services, provided by the mission critical applications utilizing FTI network services directly. When it becomes necessary for an application to share information, or a subset of it, with consumer applications, the NEMS services are utilized. The information can be published to the NEMS where it is routed, via the NAS ESB, to currently subscribed consumers. Alternately, a web service can be made available and exposed via the NEMS for consumers to request data as needed. Consumers can determine what services and content are available by accessing the NAS Service Registry Repository (NSRR). Service management services are utilized to monitor health and status for NEMS services, as well as monitor infrastructure hardware and software components of the NEMS.

Service level security ensures access to all NEMS services is limited to authorized users and provides a consolidated approach to security policy management. The NEMS enables a robust SOA security gateway, utilizing the NAS Enterprise Security Gateway (NESG), offering authorized external NAS users access to an un-trusted ESB (UESB), to enable subscriptions and web service access to NAS information. The existing security controls sit between the trusted and un-trusted ESBs protecting the trusted ESB from unauthorized access or malicious attacks while allowing NAS authorized public data to be pushed from the trusted ESB to the UESB for subsequent access by authorized external users. The security controls also allow authorized external users to publish data into the NAS as well as provide web services that can be accessed by authorized internal NAS users.



**Figure 3-1. NEMS Logical Architecture View**

The net-centric deployment approach for implementing the NEMS SOA utilizes a functionally uniform set of highly scalable and extensible NEMS nodes that can be deployed incrementally within the FTI network infrastructure as NAS applications are service enabled and deployed within the net-centric NAS. This allows the NAS to deploy processing resources only when and where they are needed, providing an effective utilization of resources over time. As depicted in Figure 3-2, there are 3 different types of NEMS nodes described in the following paragraphs. A NEMS node is a NAS-configured version of the commercial DEX platform. Also, shown in the figure is a mapping of the NEMS infrastructure services (shown in red), introduced in the NEMS logical architecture, to each NEMS node type.



**Figure 3-2. NEMS Deployment Architecture View**

The **NEMS Messaging Node** provides all messaging services, accessible to authorized NAS producers and consumers, for the NEMS. NEMS Messaging Node processors host ESB components that provide the enterprise domain tier ESB to support all access by producers and consumers to NEMS services. Messaging node processors also host security policy enforcement point components and policy driven enterprise management agents. Highly available messaging is accomplished by utilizing redundant processors.

The **NEMS System Management Node** provides all system management services which are integrated with existing FTI network service management capabilities to effectively manage the NEMS net-centric system. The system management services lend themselves to a consolidated deployment approach within the FTI NOCC using existing FTI network service management infrastructure and supporting personnel. The system management services are necessary for effective management of operational capabilities and are designed such that in the event of a failure of any of these services there is no impact to ongoing operational information exchange.

The **NEMS Access Management Node** provides access management and control services including identity management, access control management, and subscription management. These access management services are required to support operational capabilities, and in the event of a failure of any of these services, there would be a potential impact to ongoing operational information exchange. Therefore, the NEMS Access Management Node architecture is designed to be highly available, via a redundant processor approach.

The deployment status of NEMS Nodes as of 2013 is shown in Figure 3-3. NEMS node deployment at the remaining ARTCCs is expected to continue at a rate of 4 additional nodes per year.

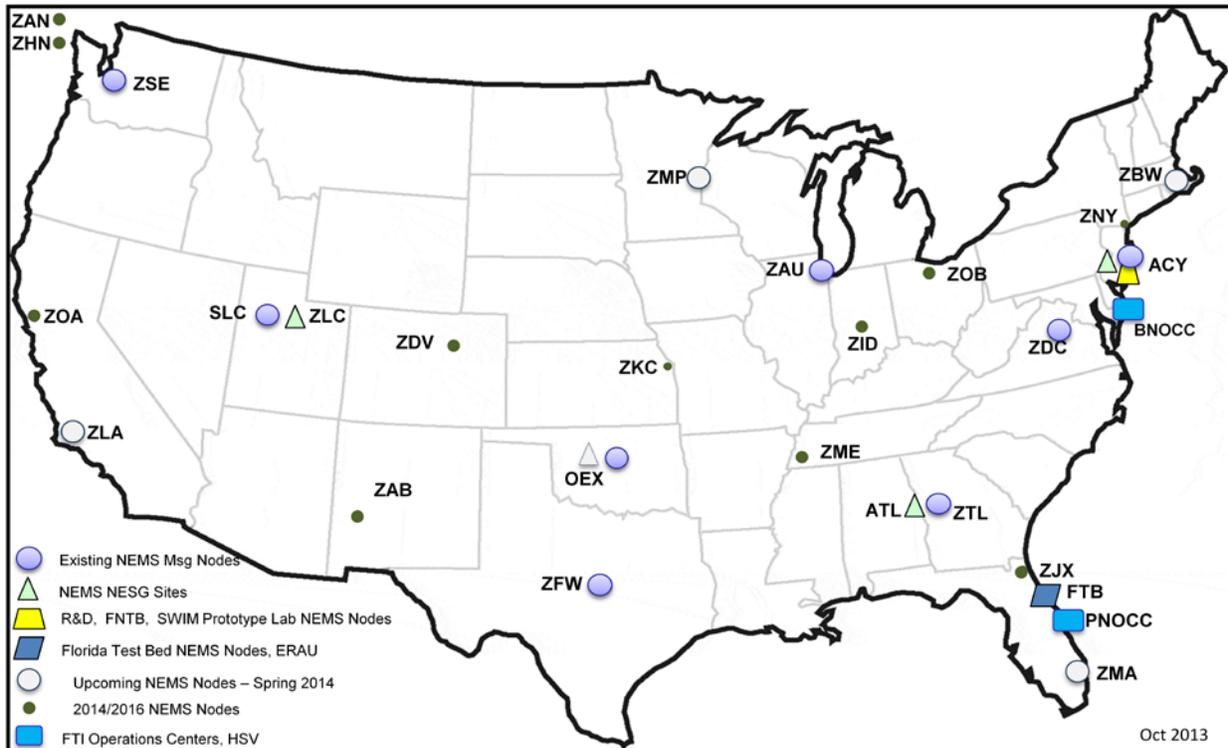


Figure 3-3. NEMS Current Deployment Status

## 4 INTERFACE DESIGN CHARACTERISTICS

### 4.1 General Characteristics

The NEMS provides net-centric core services enabling producers and consumers to exchange information within the NAS as well as to external customers. The NEMS currently supports publish/subscribe information exchange via the Java Message Service (JMS) and request/response information exchange via Restful Web services and Web Services utilizing Simple Object Access Protocol (SOAP) messages. JMS messaging services are available via Oracle WebLogic and Redhat ActiveMQ (AMQ) based NEMS brokers.

The NEMS provides mediation capabilities to transform message content and data exchange protocols.

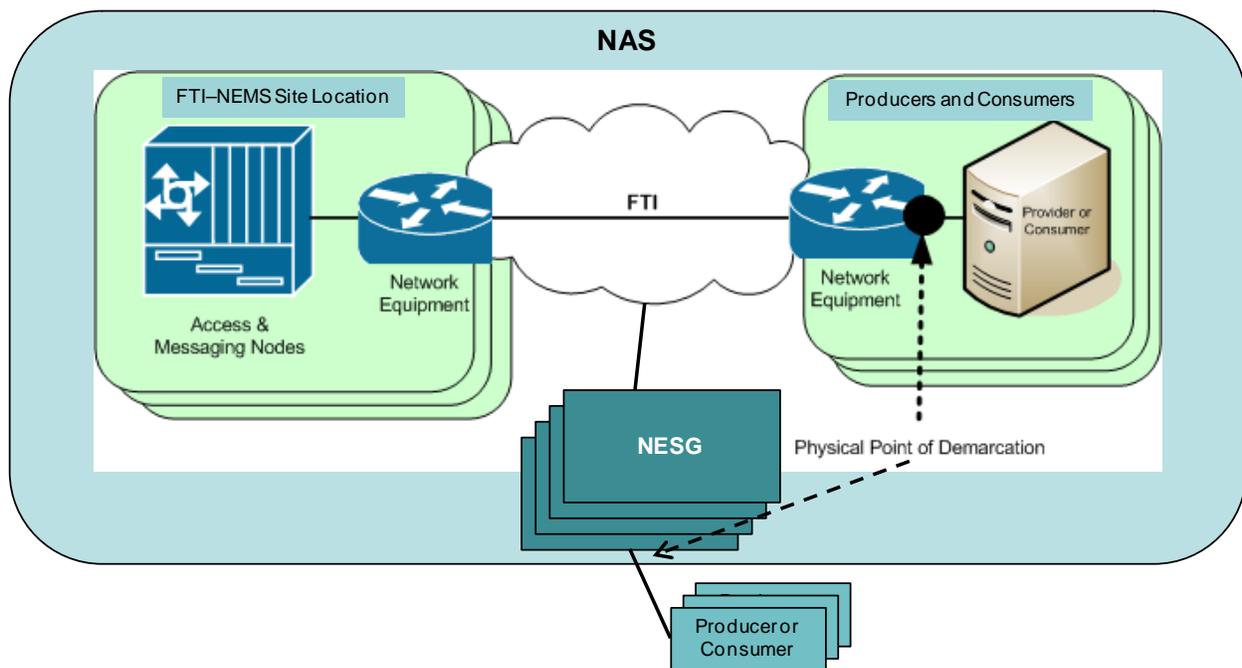
The NEMS also provides the ability to route information based on its content to applicable consumers based on configurable routing policies.

Policy based service-level security is deployed to protect NEMS services from unauthorized access with attempted security violations monitored and alarmed via configurable policies at the FTI SOCC. Access to NEMS services is restricted to authorized users with valid credentials. The health and status of NEMS core services and producer and consumer data flows are monitored and alarmed via configurable policies at the FTI PNOCC.

## 4.2 Network Interface Characteristics

For NAS users, the physical point of demarcation for access to NEMS messaging services is the Service Delivery Point (SDP) of the NAS Ops IP Network for the producer/consumer as shown in Figure 4.2-1. FTI Network health and status is monitored up to and including the FTI access router providing the user IP SDP. Access to the FTI operational network is controlled through the allocation of Internet Protocol (IP) addresses to each accessing processor. Information about NAS Ops IP services can be found in the NAS Operational IP User Guide.

For external (non-NAS) users of NEMS, the physical point of demarcation is the NESG. FTI monitors the NESG health and status including all external connections to the NESG. Access to the NESG is controlled through the allocation of Internet Protocol (IP) addresses to each accessing processor. Information about NESG services can be found in the FTI NAS Boundary Protection System (NBPS) User's Guide - Volume II for Non-NAS Users.



**Figure 4.2-1. FTI-NEMS Network Interface**

Table 4.2-1 identifies the network protocols supported by the NAS Ops IP Network and NEMS at each of the Open Systems Interconnection Reference Model (OSI) layers.

**Table 4.2-1. FTI-NEMS Network Interface Protocols**

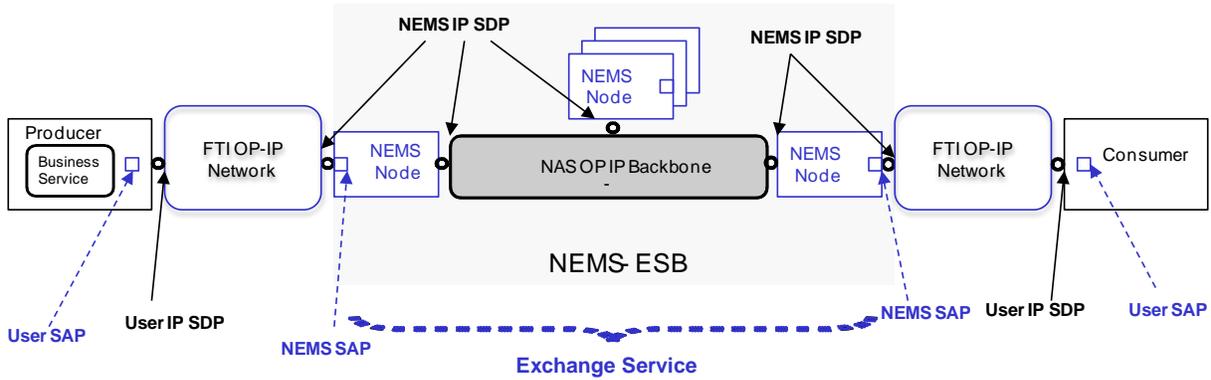
OSI LAYER	PROTOCOL
Application	JMS, HTTP, SOAP, FTP, SMTP, SNMP, NTP
Presentation	Transport Layer Security (TLS)
Session	
Transport	Transmission Control Protocol (TCP)
Network	Internet Protocol (IP)
Data Link	802.3 Ethernet
Physical	10BaseT (ISO/IEC 8802-3) and 100BaseT (IEEE 802.3u) and 1000BASE-T 1Gbps (IEEE 802.3ab)

### 4.3 Software Interface Characteristics

As shown in Figure 4.3-1, each producer and consumer connected into NEMS has its own set of User IP (Layer 3) SDPs and User Messaging (Layer 7) Service Access Points (SAPs). The User IP SDP reflects the demarcation point between the user and the NAS Ops IP Network. The User SAP represents a logical demarcation between a NEMS user (i.e. a Producer or a Consumer) and the NAS ESB for a Layer 7 application service. NEMS SAPs are allocated to producers and consumers for access to NEMS messaging services.

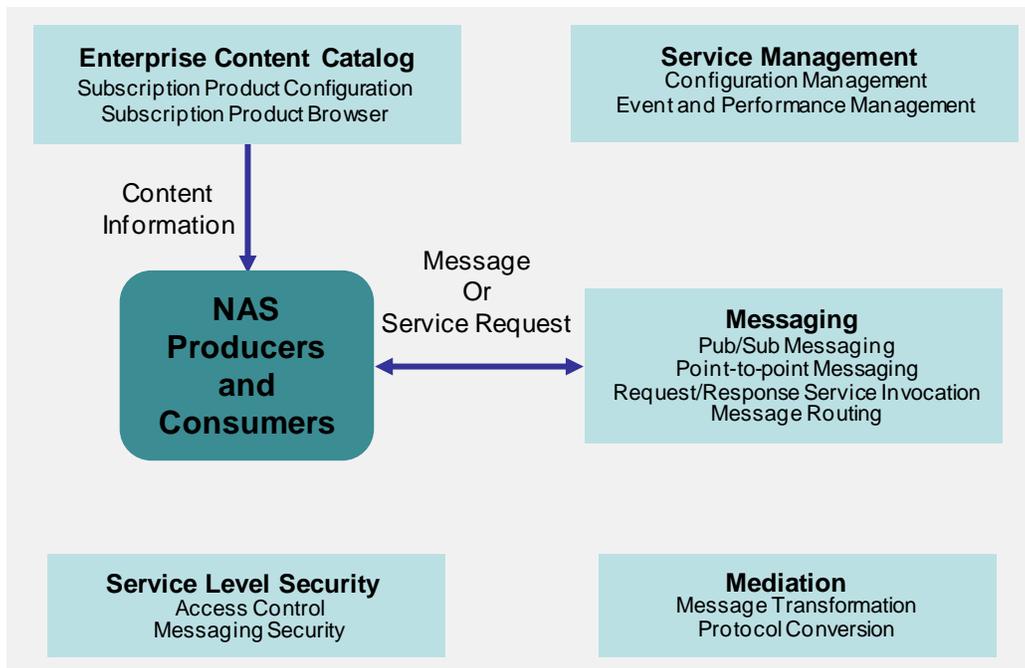
In the figure, a producer is defined as an executable process running on a server that provides NAS domain business services and associated products for access by consumers via NEMS. A consumer is defined as an executable process running on a server that accesses business services and associated products via NEMS.

An exchange service brokers producer business services and products to consumers and decouples producers and consumers. The exchange service boundaries are defined as the NEMS SAP where the message enters the NEMS infrastructure from the producer (i.e. the Producer's SAP) to the NEMS SAP where the message leaves the NEMS infrastructure to the consumer (i.e. the Consumer's SAP). The NEMS SAP is the NEMS terminating end of a Layer 7 NEMS On-ramping Service association and the User SAP is the user terminating end of a Layer 7 NEMS On-tramping Service association. The User IP SDP is the demarcation between the user (producer/consumer) system and the NAS Ops IP Network, while the NEMS IP SDP is the demarcation between the NAS ESB node and the NAS Ops IP Network. NEMS Service management tracks the association between User Messaging (Layer 7) SAPs and User IP (Layer 3) SDPs.



**Figure 4.3-1. Service Access Points (SAPs) for NEMS Services**

There are five NEMS infrastructure service categories, as shown in Figure 4.3-2. Each service category and associated usage guidance is described in the following paragraphs. The NEMS exposes interfaces to producers and consumers for the Messaging and NEMS Content Catalog services only.



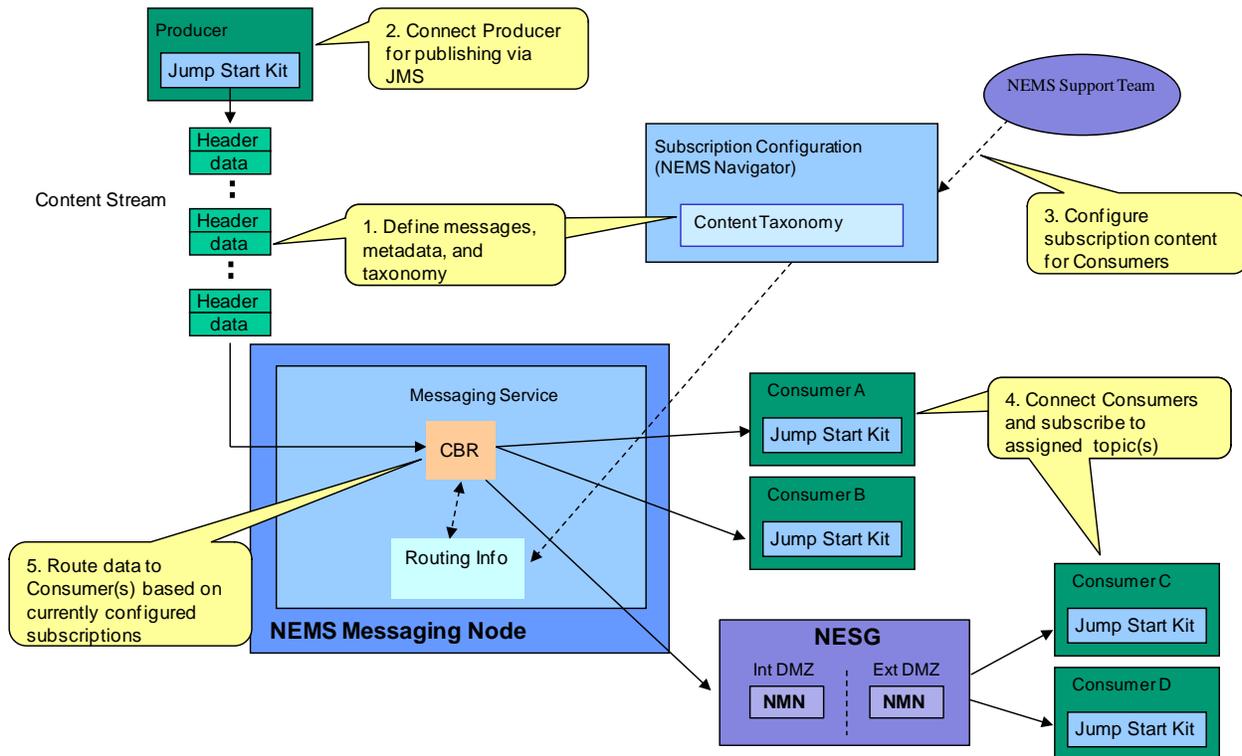
**Figure 4.3-2. NEMS Service Categories**

### 4.3.1 Messaging Service

The Messaging Service supports request/response and publish/subscribe messaging patterns. Two publish/subscribe messaging implementations are supported.

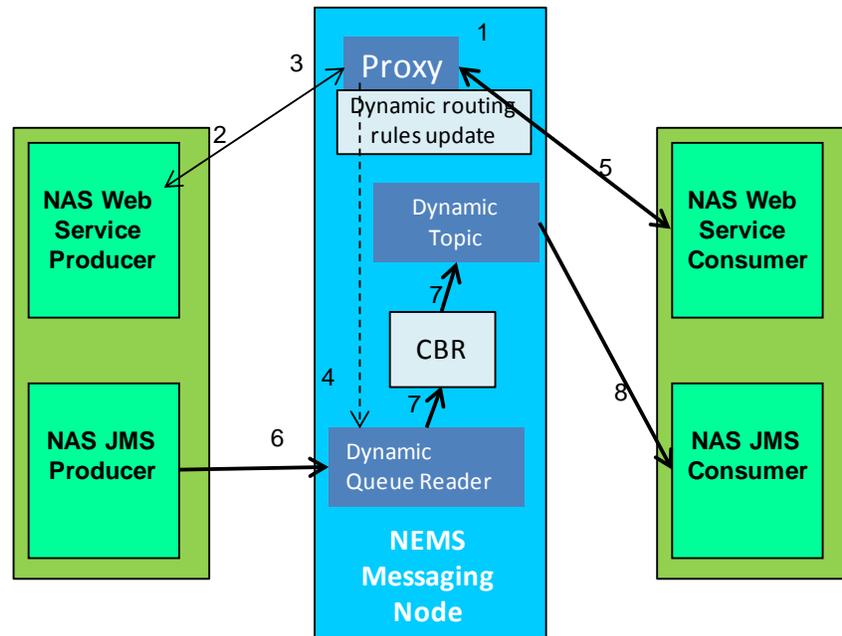
The first publish/subscribe messaging implementation, shown in Figure 4.3.1-1, is called static subscriptions and is provided using standards-based JMS messaging services. WebLogic JMS

and Active MQ JMS interfaces are provided for producers and consumers. JMS static destinations are used by producers to publish content into the NEMS. JMS messages typically contain metadata in the message header that is provided by the producer. This metadata is used to support subscription filtering and message routing to the appropriate subscribers. JMS static destinations are used by consumers to receive the content matching their subscription. The consumer subscriptions are configured by the NEMS support team as discussed in Section 5.



**Figure 4.3.1-1. Static Subscriptions Messaging**

The second publish/subscribe messaging implementation, shown in Figure 4.3.1-2, is referred to as Dynamic Subscriptions and is provided using a combination of web services and standards-based JMS messaging services. Web services are invoked to subscribe to consumer specified content. The delivery of notifications associated with the content is delivered using JMS messaging. JMS dynamically created destinations are used by producers to publish content into the NEMS. The notifications contain either the content directly or only metadata for the content. When the notifications contain only metadata, the content is then accessed by the consumer via a web service supported by the producer. This latter approach is typically used for large data sets for which it may not always be desirable to retrieve every update depending on the metadata information. This latter approach also saves network bandwidth and unnecessary processing of unwanted content. JMS dynamic destinations are created by NEMS and accessed by consumers in order to receive the content that matches their subscription request parameters.



1. WS consumer requests content via NEMS proxy.
2. NEMS proxy forwards request to business web service. Business web service creates a dynamic queue used for JMS publishing.
3. Business web service returns subscription ID and URL of queue to internal NEMS proxy.
4. Dynamic queue reader is created to read producer queue using queue URL.
5. NEMS proxy returns subscription ID and URL for NEMS topic to WS consumer. JMS consumer uses URL for connecting to topic on NEMS node.
6. JMS producer publishes message to local dynamic queue.
7. Dynamic queue reader reads message from queue and forwards to CBR. CBR routes to local topic using information from JMS header.
8. NEMS delivers message to JMS consumer. Depending on the producer implementation, the message can contain notification information about content that has been updated or can contain the actual content. If the message contains notification information only then the consumer would then retrieve the content via a web service supported by the producer.

**Figure 4.3.1-2. Dynamic Subscriptions Messaging**

Request/response messaging, shown in Figure 4.3.1-3, is provided using either SOAP or Restful Web Services. The NEMS proxies each web service hosted by a producer for access by a consumer. The producer Web Services endpoint is configured in the ESB of the Messaging service and the proxy service endpoint is made available to consumers.

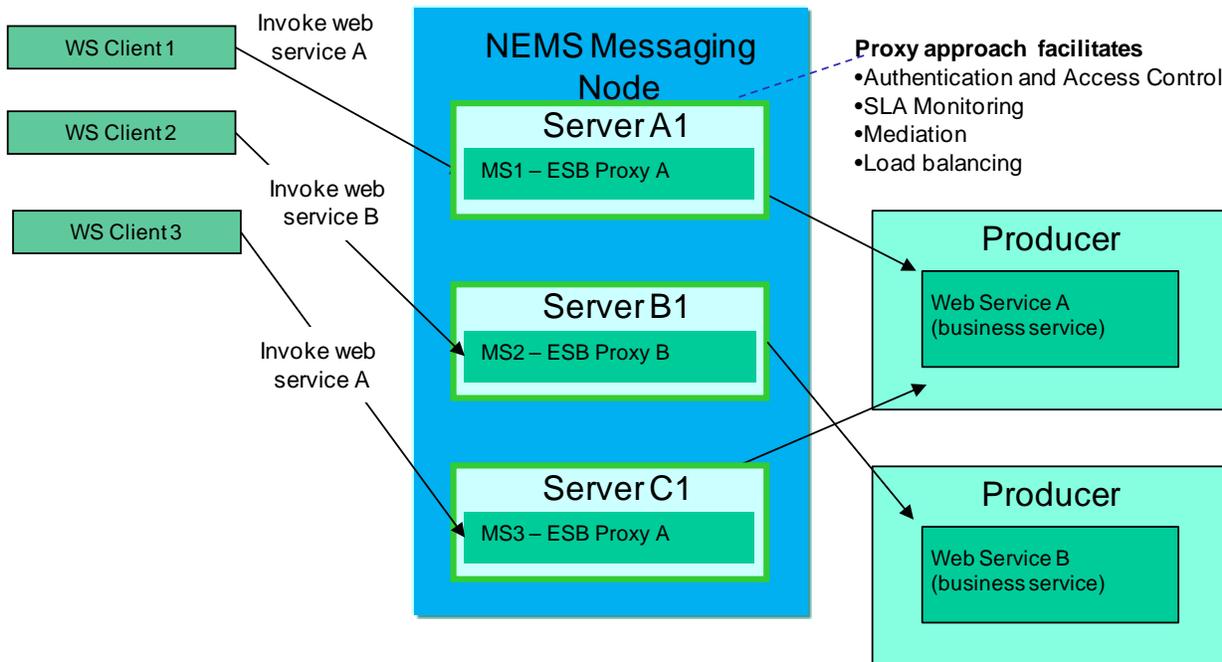


Figure 4.3.1-3. Web Service Messaging

#### 4.3.1.1 Messaging Usage Guidance

##### Publish/Subscribe

WebLogic JMS is the recommended JMS service to use for users not already using AMQ as increased throughput and lower latency performance can be accomplished. This performance increase is due to the avoidance of the AMQ to WebLogic JMS messaging bridge. The latency incurred using the messaging bridge is approximately 10 milliseconds.

The initial context binding Uniform Resource Locator (URL) is required for accessing name-to-object binding services such as Java Naming and Directory Interface (JNDI), for looking up the WebLogic JMS connection factory. Refer to the access steps in the NEMS Administration Process, Section 5, for instructions on obtaining this information. Since a JNDI is not provided by AMQ JMS, connection information for the AMQ broker is provided.

All JMS messaging by producers and consumers is performed using JMS client software. Producer-hosted broker to NEMS-hosted broker JMS message exchanges are not supported by NEMS.

JMS messages are currently delivered using a best effort quality of service (QoS). This means that data delivery is not guaranteed. Message loss is possible but all efforts are taken to reduce the risk of this event. JMS guaranteed delivery and durable subscriptions are currently not supported so in the event of a consumer failure, messages delivered during any consumer outage are lost. These QoS capabilities will be provided in planned future upgrades. However, the NEMS provides highly available messaging using multiple processors at each NEMS Messaging

Node. In the event that a producer or consumer loses a JMS connection, an attempt to reconnect can be made to another NEMS Messaging Node processor to continue receiving data.

NEMS JMS services are deployed in the NESG (implemented via 2-Way SOA Project) and support bi-directional message exchange between NAS internal and external producers and consumers.

A Data-Exchange JumpStart Kit is provided that can be used as a starting point for developing NEMS JMS producers and consumers using either WebLogic or AMQ JMS. Refer to Appendix A for Data-Exchange JumpStart Kit details.

The following standards are supported by the NEMS Messaging Service for publish/subscribe messaging.

- JMS 1.1

### **Request/Response**

SOAP and Restful services over Hypertext Transmission Protocol (HTTP) and HTTP Secured (HTTPS) are currently supported. Web Services are accessed by consumers via the NEMS. NEMS proxy Web Services are configured in the ESB of the Messaging service to provide access to the target business Web Service. The proxy Web Service is accessed by the consumer per the endpoint defined in the Web Service Description Language (WSDL) provided by the NEMS. Upon invocation, the NEMS Proxy service invokes the target business Web Service's end point passing through the original consumer request and then returns the response to the consumer. The proxy Web Service WSDL is required for accessing a business Web Service via the NEMS. Refer to the NEMS Administration Process in Section 5 for instructions on obtaining this information.

SOAP Web Services are recommended for use since they are capable of supporting advanced security policies as well as other Web Service (WS) standards including WS Addressing, WS Reliable Messaging, and WS Notification.

Web service messages are currently delivered using best effort quality of service (QoS). This means that data delivery is not guaranteed in the event of an equipment failure or network communication failure. Additional QoS capabilities, via WS Reliable Messaging, will be provided in planned future upgrades.

NEMS request/response services are deployed in the NESG and support bi-directional message exchange between NAS internal and external producers and consumers.

The following standards are supported by the NEMS Messaging Service for request/response messaging.

- SOAP 1.1, 1.2
- WSDL 1.1
- HTTP 1.0,1.1

- Message Transmission Optimization Mechanism (MTOM)
- SOAP with attachments
- Open Geospatial Consortium (OGC)
- WS Notification
- WS Addressing
- WS Reliable Messaging (scheduled for 2014)

#### **4.3.2 Service Level Security Service**

Service Level Security provides authentication and authorization capabilities. Authentication is supported using a NEMS LDAP configured as part of the on-ramping process. Role-based access control is supported via a policy based authorization component configured as part of the on-ramping process. For JMS services, authentication is required upon connection to the NEMS. Authorization is performed with respect to access to JMS resources and the operations performed with them (read, write, delete). Producers and consumers are restricted to access only assigned resources in order to provide strict access control to content. Supported JMS security credentials are username/password. Transport Level Security (TLS) connections using JMS will be provided in a planned future upgrade.

For web services, access to specific Web Services is restricted to authorized users. Authentication and authorization is provided by the web service proxy upon access. Supported security credentials for SOAP services are WS Security username token (plain). For Restful services authentication is currently provided at the HTTP layer only. SOAP security credentials can be forwarded to the business web service to enable further authentication/authorization if necessary. If the business web service is not supporting security credentials, the NEMS can be configured to remove them. Support for TLS, additional WS Security token types, and forwarding of Restful security credentials will be provided in a planned future upgrade.

SOAP and Restful web services are also supported using HTTPS.

##### **4.3.2.1 Service Level Security Usage Guidance**

User security credentials are required for accessing the NEMS, its JMS services, and Web Service proxies. Refer to the access steps in the NEMS Administration Process Section 5 for instructions on obtaining this information.

#### **4.3.3 Mediation Service**

The Mediation Service provides capabilities for protocol and message format/content transformations between producers and consumers. These capabilities are provided via configuration of the ESB.

The NEMS supports the following protocol transformations.

1. WS SOAP HTTP producer to
  - a) WS RESTful HTTP consumer
  - b) WS Extensible Markup Language (XML) HTTP consumer
  - c) WS SOAP over JMS consumer
2. WS RESTful HTTP producer to
  - a) WS SOAP HTTP consumer
  - b) WS XML HTTP consumer
3. WS XML HTTP (this is a web service that uses XML for the message format, not SOAP or restful formats) producer to
  - a) WS RESTful HTTP consumer
  - b) WS SOAP HTTP consumer
4. WS SOAP over JMS producer to
  - a) WS SOAP HTTP consumer
  - b) WS XML HTTP consumer
5. WS XML over JMS to WS XML over HTTP consumer
6. WS SOAP over HTTP to JMS publish/subscribe

Additional transformations using secure protocols will be provided in a planned future upgrade.

NEMS also provides message format and content transformations using XQuery and XSLT standard technologies to specify the desired transformation. These transformations are applied as part of the message processing pipeline as a message is processed on the ESB. The following capabilities are provided.

- XQuery or XSLT Transformations
- Textual Non-XML to XML
- XML to textual Non-XML
- XML to different XML

The following standards are supported.

- Extensible Stylesheet Language Transformations (XSLT) 1.0
- XQuery 1.0
- XPath 2.0

#### **4.3.3.1 Mediation Usage Guidance**

XQuery typically has better performance than XSLT.

Transformations should not include extensive domain specific logic to avoid tight coupling between infrastructure and business services. Transformations should not require complex operations requiring significant processing cycles as this can result in additional latency to message throughput since transformations are performed "on the wire".

Transformations can be developed by the producer organization, the consumer organization or the NEMS development organizations and can be decided on a case by case basis.

#### **4.3.4 Subscription Catalog Service**

The Subscription Catalog service provides the ability to configure NEMS subscription services that are available for subscription by consumers. Producer organizations are required to provide specific metadata required for describing their provided products in the NEMS Content Catalog. Some of this metadata is also used to define subscription granularity and is used for content based routing of content to current subscribed consumers.

The Subscription Catalog service also provides the ability for consumers to view products in the NEMS Content Catalog, available via subscription services, utilizing the NEMS Navigator.

##### **4.3.4.1 Subscription Catalog Usage Guidance**

The NEMS Navigator is a web-enabled service that provides the ability for users to view available content products in the NEMS Content Catalog and view their subscription services. The NEMS support team utilizes the NEMS Navigator to configure NEMS subscription services. Access to the NEMS Navigator by authorized producers and consumers for viewing available subscription content is also provided.

Refer to the producer and consumer registration process in Section 5 for information on registering for access to NEMS products and services including use of the NEMS Navigator.

## **5 NEMS ADMINISTRATION PROCESS**

### **5.1 Network Access Steps**

Prior to deploying a new producer or consumer, the SWIM Program Office will need to work with the FTI Program Office to determine if new FTI network access services are needed or if existing services will meet performance needs (e.g. bandwidth). If new services or changes are needed, an FTI Service Order must be placed by the FAA to provide the required network access. The order will result in the installation of any required hardware and/or cabling at the customer premise, configuration of the equipment, and provisioning of new telecommunications circuits or allocation of additional bandwidth on existing circuits.

Information about FTI IP services can be found in the FTI Operational IP User Guide.

### **5.2 NEMS Access Steps**

This section describes the steps for obtaining the information and credentials for connecting to the NEMS. It does not attempt to provide any information on the service ordering process which is managed by the FAA SWIM Program Office. Information for accessing the APIs for NEMS services via the NESG are identical to the steps defined here. The additional information for connecting to the NESG can be found in the FTI NAS Boundary Protection System (NBPS) User's Guide - Volume II for Non-NAS Users.

All NEMS services and associated data content is managed by the FAA SWIM Program Office and advertised via the NSRR. The NSRR provides a NEMS user information regarding the available services and information about the content the services provide. For NEMS subscription service content, once the content of interest has been found in the NSRR, a consumer can optionally view a list of content types and associated metadata available for subscription by using the NEMS Navigator. Refer to paragraph 5.3 for detailed instructions for using the NEMS Navigator.

The Harris NEMS support team provides security credentials verbally via phone or documented in the NEMS Connection Information form and sent via encrypted email. The NEMS support team also provides connection information, via the NEMS Connection Information form, to the consumer for access to the NEMS. Credentials can be assigned for access to one or more NEMS services.

A Data-Exchange JumpStart Kit for JMS publish/subscribe messaging using WebLogic and AMQ is provided for optional usage by a producer or consumer. Refer to Appendix A for further information regarding the Data-Exchange JumpStart Kit.

#### **5.2.1 Publish/Subscribe Consumer Steps**

The following paragraphs describe the steps for connecting a publish/subscribe consumer to the NEMS including the configuration of the subscription.

The NEMS support team provides connection information and user security credentials for accessing the NEMS Navigator.

Using the NEMS Navigator, the consumer determines the exact content that is desired for their subscription. By default, only one subscription service (JMS destination) is provided to receive as much content as required. However, more than one subscription service can be configured if needed by contacting the NEMS support team. This may be the case if the desired content set has significantly different characteristics with respect to size and frequency. If more than one subscription service is utilized, then a mapping of the selected content to each subscription service is performed by the NEMS support team in conjunction with the consumer. This information is documented using a NEMS Consumer On-Ramping form which can be obtained from the SWIM Program Office.

The NEMS support team provides security credentials verbally via phone or documented in the NEMS Connection Information form via encrypted email. The NEMS Support team also provides connection information, to access authorized NEMS subscription services, via the NEMS Connection Information form, to the consumer for access to the NEMS.

The consumer utilizes the JMS destination names in their consumer application or adds them to the configuration file of the DJK if it is being utilized for NEMS access. At this point the consumer has all required information to connect to the NEMS.

### **5.2.2 Request/Response Consumer Steps**

The following paragraphs describe the steps for connecting a request/response consumer to the NEMS.

The consumer provides a list of business Web Services to which they require access. These web services were selected using the NSRR. This information is documented using a NEMS Consumer On-Ramping form.

The NEMS Support team configures the security access control enabling access to the NEMS proxy Web Services by the consumer. The NEMS support team provides security credentials verbally via phone or documented in the NEMS Connection Information form and sent via encrypted email. The NEMS support team also provides service endpoint addresses of the proxy web services, via the NEMS Connection Information form, to the consumer for access to the NEMS.

The consumer utilizes the Web Service WSDLs, obtained from the NSRR, and the security credentials in their consumer application to access the desired business.

### **5.2.3 Publish/Subscribe Producer Steps**

The producer provides a list of content and associated descriptive metadata that will be published to the NEMS. This information is documented using a NEMS Producer On-ramping form.

The producer, working with NEMS and SWIM system engineers, defines the taxonomy that will represent the available subscription content and defines attributes in the message header to support content-based routing services provided by the NEMS.

The producer, working with NEMS and SWIM system engineers, determines whether multiple JMS queues are required for publishing to the NEMS and if so, provides a mapping of published content to JMS queues.

The NEMS support team configures the requested publishing JMS destinations and provides the JMS destination names to the producer.

The NEMS support team updates the NEMS Content Catalog with the newly published products.

The NEMS support team provides security credentials verbally via phone or documented in the NEMS Connection Information form via encrypted email. The NEMS support team also provides connection information, to access authorized NEMS JMS destinations, via the NEMS Connection Information form, to the producer for access to the NEMS.

The producer utilizes the JMS destinations names and security credentials in their producer application or adds them to the configuration file of the DJK if it is being utilized for NEMS access. At this point the producer has all required information to connect to the NEMS.

#### **5.2.4 Request/Response Producer Steps**

The producer supplies a list of Web Services and associated descriptive metadata (e.g. WSDD) and WSDLs to be made accessible via the NEMS. This information is documented using a Producer On-ramping form. These WSDLs are also registered in the NSRR.

A proxy for each producer web service is developed and deployed, by the NEMS support team, to the ESB of the Messaging service.

#### **5.2.5 Producer Disconnect**

Producers provide a notification for a service or content that is being removed from service.

The NEMS support team performs configuration and deployment tasks necessary to decommission a service or content from the NEMS.

#### **5.2.6 Consumer Disconnect**

Consumers provide a notification for a service or content that is no longer being consumed.

The NEMS support team performs configuration and deployment tasks necessary to terminate delivery of the indicated services and content to the consumer.

### 5.3 NEMS Navigator Access Instructions

The NEMS Navigator provides the capability to view content available for subscriptions but does not allow the user to configure a subscription. The actual subscription configuration is done by the NEMS support team.

The following steps are taken to view subscriptions via the NEMS Navigator.

1. Access the NEMS Navigator at the addresses provided by the NEMS Connection Information Form. There are separate addresses for internal and external NAS users.
2. Enter your assigned user-id and password.
3. Select the *Products* tab for a list of available products (this will be the default tab upon login). Products appear on left side pane and are arranged in a hierarchical taxonomy. Click on arrows to navigate product hierarchies.
4. Navigate through the product list to determine the required products.

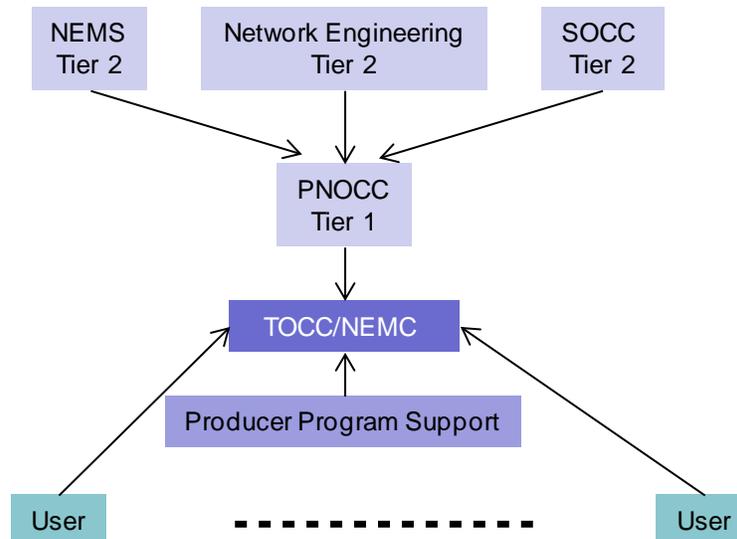
## 6 NEMS DEVELOPMENTAL SUPPORT

Harris provides engineering support to the SWIM program for all NEMS on-ramping activities. This includes working in concert with the SWIM program to assist prospective producers and consumers in determining the optimal approach for on-ramping services and content to the NEMS and making them available to consumers. This includes determining the applicable messaging patterns, defining subscription content taxonomies, defining message meta-data used for routing, understanding security policies and their implementation, determining service monitoring strategies, configuring required qualities of service, and solving other technical issues that may arise during the on-ramping process.

Requests for Harris support during on-ramping developmental activities should be initiated through the SWIM program office so that all support activities can be effectively coordinated.

## 7 NEMS OPERATIONAL SUPPORT

NEMS provides 24/7 centralized infrastructure management and support. As shown in Figure 7-1, NEMS users contact the FAA's National Enterprise Management Center (NEMC) which provides 24x7 tier one support as well as monitoring and control of critical network and application functions. The NEMC will contact the FTI support desk as needed for any NEMS specific support or will contact the producer program support organization for any producer specific support.



**Figure 7-1. Operational Support Organization**

The NEMS infrastructure is owned and operated by Harris Corporation, the prime contractor. Harris is responsible for managing all maintenance activities for the NEMS. These activities include performing scheduled maintenance, restoration, repair, and configuration of the NEMS equipment and software, as well as the FTI network services used by NEMS.

The focal point for managing these activities is the FTI Primary Network Operations Control Center (PNOCC) in Melbourne, Florida; a Backup NOCC (BNOCC) is located in Chantilly, Virginia. The FTI NOCCs have a single POC as far as the FAA is concerned, for both NEMS and FTI services; this is referred to as the PNOCC (with the understanding that the BNOCC may actually be the active control center at times). In general, maintenance functions performed by the FTI NOCC include:

- Providing oversight and management of the FTI network and NEMS infrastructure
- Remotely monitoring and controlling installed equipment
- Coordinating the activities of partner service providers (as applicable)
- Coordinating scheduled maintenance activities
- Coordinating response to unscheduled service interruptions and/or outages
- Exchanging information about the status, configuration, and performance of NEMS services to various Technical Operations organizations.

FTI (and NEMS) uses a Unique Service Identifier (USI) to designate each specific interface, including the layer 7 NEMS services interfaces. Each USI has a designated primary and backup Technical Operations Control Center (TOCC) contact (e.g., a NEMC). When the PNOCC needs to contact the FAA about a specific USI (e.g., to report a problem or to request maintenance release), the PNOCC will contact the TOCC. Likewise, users should contact their TOCC

regarding operational issues with NEMS services. Users and/or TOCCs identify their USIs while communicating with the PNOCC for easy identification and better service.

## 8 NOTES

### 8.1 Abbreviations and Acronyms

The following list of acronyms and definitions are used throughout this guide and are applicable to the FTI Program.

**Table 8.1-1. Acronyms and Definitions**

<b>ACRONYM</b>	<b>DEFINITION</b>
AMQ	Active MQ
AIXM	Aeronautical Information Exchange Model
ANSP	Air Navigation Service Provider
API	Application Programming Interface
ARTCC	Air Route Traffic Control Center
BNOCC	Backup Network Operations Center
CAGE	Commercial and Government Entity
CBR	Content Based Routing
CDRL	Contract Data Requirements List
DCN	Document Control Number
DEX	Data Exchange
DHS	Department of Homeland Security
DJK	Data-Exchange JumpStart Kit
DMZ	Demilitarized Zone
DoD	Department of Defense

**Table 8.1-1. Acronyms and Definitions (Continued)**

ACRONYM	DEFINITION
ESB	Enterprise Service Bus
FAA	Federal Aviation Administration
FIXM	Flight Information Exchange Model
FNTB	FTI National Test Bed
FTI	FAA Telecommunications Infrastructure
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ID	Identifier
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical And Electronic Engineers
IP	Internet Protocol
ISO	International Organization for Standardization
JMS	Java Message Service
JNDI	Java Naming and Directory Interface
LAN	Local Area Network
LDAP	Light Weight Directory Access Protocol
MQ	Message Queue
MTOM	Message Transmission Optimization Mechanism
NAS	National Airspace System
NBPS	NAS Boundary Protection System
NEMC	Network Enterprise Management Center

**Table 8.1-1. Acronyms and Definitions (Continued)**

ACRONYM	DEFINITION
NEMS	NAS Enterprise Messaging Services
NESG	NAS Enterprise Security Gateway
NMN	NEMS Messaging Node
NOCC	Network Operations Control Center
NSRR	NAS Service Registry/Repository
NWP	NextGen Weather Processor
OGC	Open Geospatial Consortium
OP-IP	Operational-IP
OSI	Open Systems Interconnection
POC	Point of contact
PNOCC	Primary Network Operations Control Center
QoS	Quality of Service
R&D	Research & Development
SAP	Service Access Point
SDP	Service Delivery Point
SLA	Service Level Agreement
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SOCC	Security Operations Control Center
SWIM	System Wide Information Management
TCP	Transmission Control Protocol

**Table 8.1-1. Acronyms and Definitions (Continued)**

<b>ACRONYM</b>	<b>DEFINITION</b>
TLS	Transport Level Security
TOCC	Technical Operations Control Center
UESB	Un-trusted Enterprise Service Bus
URL	Uniform Resource Locator
USI	Unique Service Identifier
WAN	Wide Area Network
WARP	Weather and Radar Processor
WS	Web Service
WSDD	Web Services Description Document
WSDL	Web Service Description Language
WXXM	Weather Information Exchange Model
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language Transformations

**APPENDIX A**

**DATA-EXCHANGE JUMPSTART KIT PRODUCER AND CONSUMER  
OVERVIEW**

## A. INTRODUCTION

The purpose of this appendix is to provide an overview of the Data-Exchange JumpStart Kit producer and consumer capabilities that can be used to send and receive data via the NEMS. The DJK is meant to serve as an example implementation of a producer and consumer. In addition, it can serve as a framework that can be customized for a particular application by defining new input and output classes.

The DJK is available upon request and includes the following additional information: a Readme.txt file containing installation information, scripts, source code, configuration files and all other data necessary for deploying the DJK producer and consumer clients.

### A.1 Data-Exchange JumpStart Kit Producer

#### A.1.1 JProvider Overview

The DJK producer is a simple producer for publishing data to the NEMS. It is designed to be easily customized for use with a number of different data sources.

The job of the producer is to obtain data and publish it to the NEMS for distribution to NEMS consumers. Data can be obtained from any source such as a file, a directory of files or a URL. The logic necessary for obtaining the data is separate from the logic that creates and sends the messages, so the implementation details of reading in data can be easily changed.

#### A.1.2 JProvider Classes

There are three Java classes that make up the core of the DJK producer. These are the input, output and message object.

MessagingInput is an abstract class which represents the “main” class of the producer. When the producer is run, the specified subclass of MessagingInput reads in data and creates one or more message objects, which it then passes to the output. This is the class which should be extended in order to implement custom message creation code.

MessagePayload is a class that represents a message. It contains a byte array to hold the content of the message and a map to hold name-value pairs representing message properties. Each instance of this class represents a single message to be sent.

MessagingOutput implementation receives MessagePayload objects from the MessagingInput class via a MessagingEvent object and processes them based on the implementation of the MessagingOutput interface being used.

A typical execution of the producer will follow these steps:

1. The producer is started. An instance of whichever MessagingInput subclass is specified in the config file is created.
2. The input class's start() method is then called.

Each `MessagePayload` is passed to a `MessagingOutput` implementation via a `MessagingEvent` object by calling the `fireMessageCreated()` method of the `MessagingInput` object.

As the `MessagingOutput` instance receives each `MessagePayload` object, it reads the message content and properties, and processes the message based on the implementation-specific functionality of the `MessagingOutput` instance. This could include creating a `BytesMessage` object from the message (`JMSOutput`), or writing the messaging to a file (`FileOutput`).

### A.1.3 Custom Inputs

The `MessagingInput` class is an abstract class. An implementation of this class should receive messages from an outside source (e.g., a JMS server or files in a directory), transform the message into an appropriate `MessagingEvent` object, and forward them on to an implementation of the `MessagingOutput` interface via calls to the `fireMessageCreated()` method as follows:

```
fireMessageCreated(aPayload, Calendar.getInstance());
```

`aPayload` here should be replaced by a `MessagePayload` object, or any object that will represent a message, and will be understood by the `MessagingOutput` implementation.

An example implementation of `MessagingInput` is `FileInput`, which is included in the `com.harris.gcsd.dex.jumstart.input` package. The data source string read in by this class will represent a file or directory path. `FileInput`'s `generateMessages()` method reads in the file(s) present at the path and creates a `MessagePayload` object for each one.

When a `MessagePayload` is created, its content and properties are empty. The content can be set by calling the object's `setPayloadContent()` method, which takes as a parameter a byte array containing the desired content. Properties can be set by calling the `MessagePayload` object's `setProperty()` method, which accepts two strings as parameters: the first specifies the property name and the second specifies that property's value.

The properties that need to be present on each message, and their values, will be determined by routing requirements and application requirements. The NEMS will require certain properties to be present on each message in order for it to be properly routed. These properties will be decided upon when the products and destinations are being configured in the NEMS system. Application-required properties are not used for routing, but may be required for proper handling of the message when it reaches its destination. For example, `FileInput` may need to set the input file's name as a property on the message so that a consumer at the destination can write the data out to a file with the same name.

In addition to the message properties, the `MessagePayload` object has an attribute called `theIdentifier`. This attribute is not included in the JMS message sent to the NEMS; rather, it is written to the producer's products log to identify each message as it is sent. For example, `FileInput` sets each message's identifier equal to the absolute path to that message's original file. This makes the products log useful for identifying each message that is sent.

Any class that extends `MessagingInput` will need to implement a constructor similar to the following:

```
public FileInput(List<MessageListener> outputs)
{
    super(outputs);
}
```

This constructor accepts a list of `MessageListener` objects which are to be registered as outputs for this class. These classes are specified in the configuration file and should not need to change.

Classes extending `MessagingInput` can also define properties which can be specified in the configuration file. To do this, simply create a java property as normal, with appropriate getters and setters. Then, the property can be initialized with an arbitrary value in the config file; this is discussed in the following section. `FileInput` defines a property called `pollingTime`, which determines how long the producer should wait between each read of the directory.

#### A.1.4 Using a Custom Input

Once a custom input has been created, it needs to be specified in the XML config file. The section of the file used to configure the input looks like this:

```
<!-- The MessagingInput object -->

<bean lazy-init="true" id="fileInput"
class="com.harris.gcsd.swim.jumpstart.provider.input.FileInput">

    <!-- The list of outputs. -->

    <constructor-arg>

        <list>

            <ref bean="jmsOutput" />

        </list>

    </constructor-arg>

    <!-- A string representing the data source, which will be passed to the processor
class. -->
```

```
<property name="dataSource" value="/opt/swim/JumpstartProvider/data/inbox/"  
>
```

```
<!-- The amount of time to wait between each poll of the data source, in  
milliseconds. -->
```

```
<property name="pollingTime" value="5000" />
```

```
</bean>
```

Modifying the producer's input consists of two steps. First, the class used by the producer should be specified on the following line:

```
<bean lazy-init="true" id="jumpstartInput"  
class="com.harris.gcsd.swim.jumpstart.provider.input.FileInput">
```

The value of the class attribute should be changed to the fully-qualified name of the input class to be used.

Second, the `dataSource` property should be modified to represent a valid data source for the input by modifying the following line:

```
<property name="dataSource" value="/opt/swim/JumpstartProvider/data/inbox/" />
```

In the above example, `FileInput` is designed to read files and directories, so the value of `dataSource` is a directory path. If a different input class is to be used, then `dataSource` should be changed appropriately.

Finally, any properties specified in the input class can be set in this part of the config file. For example, `FileInput` defines a property called `pollingTime` which is initialized by the following line:

```
<property name="pollingTime" value="5000" />
```

The property is initialized by calling a method named `setPollingTime()` in the `FileInput` class and passing it the value 5000. Any number of properties can be defined in this manner so long as the appropriate setter methods exist in the input class. In general, a property named `ABC` will be initialized by calling the input class's method `setABC()`.

Note that any changes made to the configuration file, including modifications to the input class, data source, or any properties, will take effect when the producer is restarted.

## A.2 Jumpstart Consumer

### A.2.1 JConsumer Overview

The DJK consumer is a simple consumer for subscribing to NEMS provided information. It is designed to provide an easily-customizable process for obtaining and processing data from NEMS.

The job of the consumer is to subscribe to a JMS destination, receive data from the NEMS and process it however the user desires. Typical tasks might include writing the data to disk or printing it to the screen.

### A.2.2 JConsumer Classes

The DJK consumer requires only two classes to run. The first, JMSInput, handles creating a connection to the NEMS and listening for messages on the specified destination. When a message is received, it is passed to the output class.

The output class determines how the received message is handled. When the output receives a message from JMSInput, its messageCreated() method is called, which handles reading the message and processing the contents. This class can be modified or replaced to change the behavior of the consumer.

### A.2.3 Custom Outputs

An output class must implement the MessagingOutput interface, which is part of the messaging framework. This interface specifies three methods that must be implemented: init(), close(), and messageCreated(). The init() method is called when the consumer is started, and the close() method is called just before the consumer shuts down. Any code that needs to run at one of these times can go in these methods; if no such code is necessary, they can be left empty.

The messageCreated() method is called whenever the consumer receives a message. The message is passed to the method as a MessageEvent object. The message itself can be obtained by calling the MessageEvent object's getMessage() method, which returns the message as a MessagingPayload object. The message content can then be retrieved with the MessagePayload's getContent() method, and its properties can be obtained with the MessagePayload's getProperty method.

### A.2.4 Using a Custom Output

Once a custom output has been created, it can be used by simply adding it to the consumer's configuration file. First, a bean should be created specifying the output class. This is done by adding a line to the config file in the following format:

```
<bean lazy-init="true" id="BEANNAME" class="CLASSNAME" ></bean>
```

BEANNAME here should be replaced with a string that will identify this bean. CLASSNAME should be replaced with the fully-qualified name of the output class.

It is possible to initialize a property of the output class here by including a <property> tag inside the <bean> tag. Consider the following example:

```
<bean lazy-init="true" id="fileOutput"  
      class="com.harris.gcsd.swim.jumpstart.consumer.output.FileOutput">  
  <property name="outputDirectory" value="/opt/swim/JumpstartConsumer/data" />  
</bean>
```

Here, a property named outputDirectory is initialized with a directory path. When the FileOutput class is initialized, this string will be passed to a setter corresponding to the property name. The property in this example is called outputDirectory, so the setter that will be called is setOutputDirectory(). If this setter does not exist, an error will occur.

After the output class has been specified, it needs to be added to the input's list of outputs. This section of the config file to be edited looks like this:

```
<bean id="consumerInput"  
      class="com.harris.gcsd.swim.jumpstart.consumer.input.JMSInput">  
  
  <description>Arguments for the constructor</description>  
  
  <constructor-arg index="0">  
  
    <util:list>  
  
      <ref bean="BEANNAME" />  
  
    </util:list>  
  
  </constructor-arg>  
  
</bean>
```

A string representing the output class's bean is added to the <constructor-arg> tag of the input class's bean. BEANNAME here should be replaced with the name given to the output class's bean above.

Note that although multiple output classes can be added here, each message that is received will be passed to all of them. Therefore, if multiple outputs are used, it is necessary to check the message type in each output class to ensure that each message is only handled by the appropriate output.