

**EUROPEAN COMMISSION**

**EUROPEAN MARITIME SAFETY AGENCY**

Cais Do Sodré 1249-206 Lisbon, Portugal

# SafeSeaNet System Design Document

---

SSN EIS

---

*Document version: 1.46*

*Document release date: January 2015*

*Application version :3.0.2*

*XMLRG version :3.02*

### *Document Approval*

	NAME	DATE	SIGNATURE
Prepared by:	C. Routis Y. Tassopoulos, C. Trigonis	27.01.2015	
Checked by:	A. Argyropoulos	28.01.2015	
Quality control by:	N. Karioti	28.01.2015	
Approved by:	G. Carayannis		

### *Distribution List*

COMPANY	NAME	FUNCTION	FOR APPROVAL	INFO /
EMSA				
EMSA				
Member States				
SSN central system contractor				

### *Change control History*

VERSION	DATE	AUTHOR	DESCRIPTION
0.90	25.04.2014	Intrasoft International	Submitted to EMSA for review.
0.95	23.05.2014	Intrasoft International	Updated to integrate SSN specific requirements based on SC#09 under FC 11/EMSA/OP/08/2011. Submitted to EMSA for review.
0.96	18.06.2014	Intrasoft International	Working version (processing EMSA comments).
1.00	01.07.2014	Intrasoft International	Incorporated EMSA comments.
1.10	18.07.2014	Intrasoft International	Incorporated EMSA comments. Submitted for acceptance.
1.20	06.08.2014	Intrasoft International	Corrected text according to the acceptance review by EMSA.
1.30	01.09.2014	Intrasoft International	Updated to incorporate the new version of the MRS notification in the context of SC#10.
1.35	17.09.2014	Intrasoft International	Incorporated EMSA comments in the context of SC#10. Submitted for acceptance.
1.40	22.10.2014	Intrasoft International	Incorporated final set of EMSA acceptance review comments.
1.45	14.11.2014	Intrasoft	Incorporated further comments from EMSA.

		International	
1.46	28.01.2015	Intrasoft International	Updated sections 3.1.2, 4.4 and 5 to define the ssn-xmlprotocol-app for XMLRG v2 messages support. New section 4.5 for the Central Database services.

## **Table of Contents**

1	Introduction .....	7
1.1	Purpose .....	7
1.2	Scope .....	7
1.3	Reference documents .....	7
1.4	Abbreviations and acronyms .....	7
2	Architectural Goals and Constraints .....	10
2.1	Service-Oriented Architecture (SOA) .....	10
2.2	Java EE Technologies .....	11
2.3	Industry Standards .....	11
2.4	Business Processes .....	12
2.5	Open Source Frameworks .....	12
2.5.1	Spring framework .....	12
2.5.2	Hibernate framework .....	13
2.6	Application Server .....	13
2.7	Database .....	14
2.8	Non-functional requirements .....	14
3	Overall System Architecture .....	15
3.1	Functional Architecture .....	15
3.1.1	Handle Incoming Message .....	17
3.1.2	Data Provide .....	20
3.1.3	Data Request .....	41
3.1.4	Monitoring Incident Report .....	68
3.1.5	MRS Management .....	69
4	Design of System Components .....	72
4.1	SSN-EIS .....	72
4.2	SSN Core Application - ssn-core-app .....	75
4.2.1	SSN Core Main Components .....	75
4.2.2	UML Class and Sequence Diagrams .....	82
4.2.3	Security on the ssn-core-app .....	159
4.3	SSN Console Application - ssn-console-app .....	160
4.4	SSN XML Protocol Application - ssn-xmlprotocol-app .....	160
4.4.1	SSN XML Protocol Main Components .....	162
4.4.2	UML Class and Sequence Diagrams .....	163
4.4.3	Interfaces between the ssn-xmlprotocol-app and the ssn-core-app .....	178
4.4.4	Security on the ssn-xmlprotocol-app .....	179
4.5	SSN-Central Database (CD) .....	179



4.5.1	Location Service .....	182
4.5.2	Authority Service.....	183
4.5.3	Ship Particulars Service .....	185
4.6	SSN-IMDaTe.....	187
4.6.1	Domain module/package .....	190
4.7	STIRES Core.....	193
4.8	SSN-VMS .....	195
4.8.1	Domain module/package .....	198
4.8.2	Business Support module/package.....	202
4.8.3	VMS-Proxy.....	202
4.9	SSN-Blue Belt .....	203
4.9.1	SSN Blue BeltCore Application - ssn-bluebelt-core-app .....	205
4.9.2	BB Console Application - ssn-bluebelt-console-app .....	210
4.9.3	Message Queue .....	210
4.10	SSN GI .....	210
4.10.1	Spatial Information Visualisation Scheme .....	211
4.10.2	Structure.....	212
4.10.3	Behaviour .....	213
4.11	SSN-GI / EIS notification details request protocol mechanism upgrade.....	215
4.11.1	Current mechanism .....	215
4.11.2	Proposed improvement.....	216
4.12	Message Queues .....	216
4.13	LOCODE Management – Upload LOCODEs .....	218
4.14	Vessel Management – Upload Single Hull Tankers .....	219
4.15	Vessel Management – OVR Synchronization.....	219
4.16	EIS & STIRES interoperability – Get Enrichment data.....	221
4.17	Voyage Calculation Process.....	222
5	Deployment view .....	226
5.1	Design Decisions .....	226
5.2	SSN EIS .....	226
5.2.1	EIS Console Server .....	228
5.2.2	EIS Core Server .....	228
5.2.3	EIS Resources Console Server.....	229
5.2.4	EIS Resources Core Server .....	230
5.2.5	SSN GI.....	230
5.3	SSN-IMDaTe.....	230
5.4	SSN-VMS .....	231

5.5 SSN-Blue Belt .....	233
Annex A: Business Rules.....	234
Voyage Status Indicators.....	234
Voyage retrieval specific rules .....	235

## 1 Introduction

### 1.1 Purpose

This document defines the SSN system design document. Additional documents that complete the SSN system design specification are:

- GIDD-TI: Graphical Design Document – Textual Interface
- GIDD-GI: Graphical Design Document – Graphical Interface
- Sddb: System Database Design document

### 1.2 Scope

This document is the *System Design Document* (hereinafter the SDD) for SSN system. It presents a number of different architectural views to depict different aspects of the system. The purpose of this document is to present the technical details of the system components and more specifically:

- The definition of domain entities and the methods that implement the requested functionality.
- The creation of UML Class Diagrams and UML Sequence Diagrams, which associate classes and depict the overall flow of control within the system components respectively.

The primary intended audience of this document are system designers and system builders. The document intends to provide the members of the SSN v2 project a unified view of the technical details of the system design to be followed during the development of the respective application. The document may need to be updated later to incorporate possible changes during development.

The current version of the document incorporates the design changes part of SC#09 and SC#10 under FC 11/EMSA/OP/08/2011 concerning the SSN changes to accommodate the V3 Ship, PortPlus, ShipCall and Exemptions notifications and request/response messages.

### 1.3 Reference documents

Id	Reference	Title	Version
R1	RUP Formal Resources Version 1.2	Rational Unified Process Formal Resources based on RUP Version: 2003.06.13	1.2
R2	N/A	<a href="#">Business Process Modelling Notation (BPMN)</a>	1.2
R3	SSN-EIS-SRS	SSN EIS System Requirements Specifications	0.90
R4	SSN- XMLMessagingRefGuide	SSN XML Reference Guide	3.01
R5	SSN-SRS-SC10	SC#10 under FMC EMSA 11/EMSA/OP/08/2011	1.00

Table 1-1: Reference Documents

### 1.4 Abbreviations and acronyms

A list of the principal abbreviations and acronyms used in the document is provided here for a better understanding of this document.

Abbreviation	Definition
--------------	------------

AIS	Automatic Identification Systems
EIS	European Index Server
EMSA	European Maritime Safety Agency
ER	Entity Relationship
ERD	Entity Relationship Diagram
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol over SSL
ID	Identification number
IMO	International Maritime Organisation
ISO	International Organization for Standardization
LCA	LOCAL Competent Authority
MMSI	Maritime Mobile Service Identity
MRS	Mandatory Ship Reporting System
IMDatE	Integrated Maritime Data Environment
IMO	International Maritime Organisation
ISO	International Organization for Standardization
IoC	Inversion of Control
IE	Internet Explorer
II	INTRASOFT International
ITU	International Telecommunications Commission
ITU_1	<a href="#">AIS Message Type 1</a> AIS Vessel position report using SOTDMA (Self-Organizing Time Division Multiple Access). This is the most common AIS message type.
ITU_3	<a href="#">AIS Message Type 3</a> AIS Vessel position report using ITDMA (Incremental Time Division Multiple Access).
ITU_5	<a href="#">AIS Message Type 5</a> Ship static and voyage related data. This is the third-most common AIS message type. Due to its length it is generally a 2-part message.
JMS	Java Message Service
JNDI	Java Naming and Directory Interface; used for reference to Java EE components defined in WebLogic server.
JSON	<a href="#">JavaScript Object Notation</a> is a lightweight data-interchange format
JSP	Java Server Pages

JSTL	JavaServer Pages Standard Tag Library
MMSI	Maritime Mobile Service Identity
MRS	Mandatory Ship Reporting System
MS	Member States.
MSS	Maritime Support Services
MSS Tool	Messages Availability MSS tool
MVC	Model-view-controller
N/A	Not Applicable or Not Available
NCA	National Competent Authority
OVR	Operational Vessel Registry
Req	Request
RUP	Rational Unified Process
SAT	Ship Activity Tracking
SQL	Structured Query Language
SSL	Secure Socket Layer
SSN	SafeSeaNet
TR	Table Reference
TRD	Table Reference Diagram
UML	Unified Modelling Language
URL	Unified Resource Locator
UTC	Coordinated Universal Time
V&V	Vessel Verification and Validation
XML	eXtensible Markup Language

**Table 1-2: Abbreviations and Acronyms**

---

## 2 Architectural Goals and Constraints

---

This section describes the software requirements and objectives that have some significant impact on the architecture.

The SSN architecture is based on the following architectural principles and technologies:

### 2.1 Service-Oriented Architecture (SOA)

A SOA is an architecture principle that is based on the key concept of services. A service, in its simplest form, consists of an interface and an implementation. SOA defines software applications in terms of discrete services, which are implemented using service components that can be used to perform business activities for a given business process.

Service Component Architecture (SCA) is a set of specifications which describe a model for building applications and systems using a Service Oriented Architecture. SCA models solutions as sets of service components offering services and making references to services supplied by others, which are combined together by composites which wire references to services and which declaratively apply bindings for communication methods and also apply policies for aspects such as security and transactions. SCA extends and complements prior approaches to implementing services, and SCA builds on open standards such as Web services.

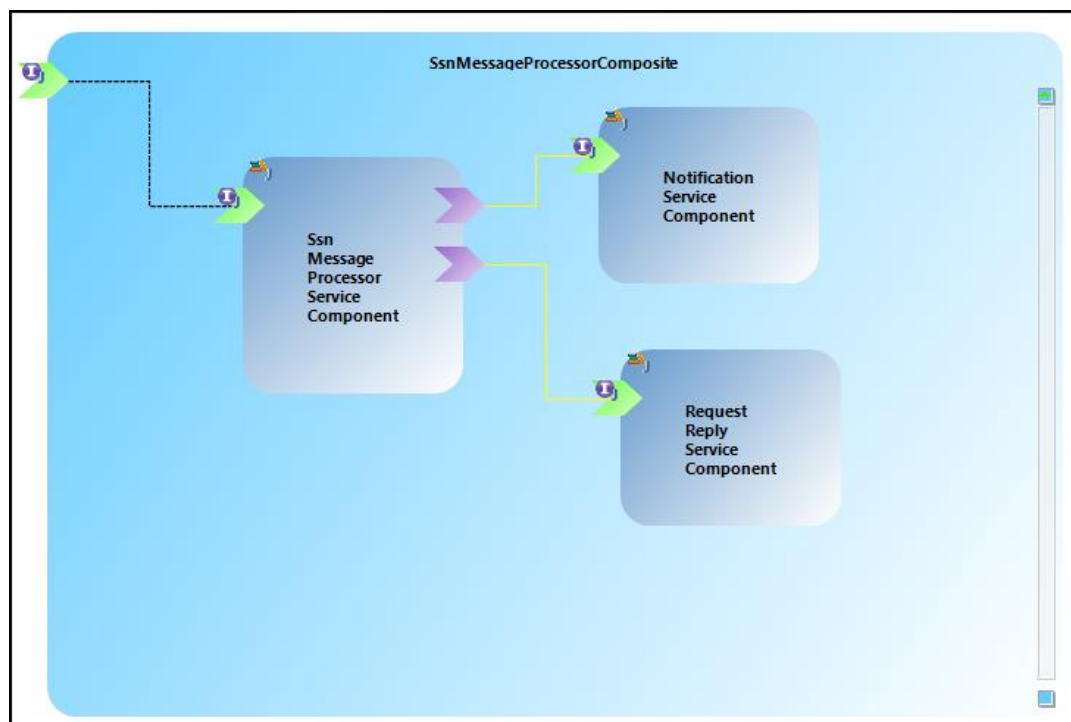
The Service Component Architecture (SCA) assembly model abstracts the implementation and allows assembly of components, with little implementation details. SCA facilitates the business logic representation as reusable service components that can be easily integrated into any SCA-compliant application. The resulting application is known as a SOA composite application.

It is possible to use an existing Spring application context as a component implementation in SCA. An SCA runtime (Weblogic SCA container) that supports Spring integration can use an application context as-is in an SCA assembly. For such a component it is possible to wire Spring services and references without the need to introduce SCA metadata into the Spring configuration. The Spring context needs to know very little about the SCA environment. Two points where the SCA metadata interacts with the Spring context are services and references. Any policy enforcement such as the provision of Security features is done by the SCA runtime on calls into the Spring application context before the final message is delivered to the target Spring bean. On outbound calls from the application context, references supplied by the SCA can provide policy enforcement.

It is also possible to specify SCA-related metadata as beans inside a Spring configuration. The Spring Component Implementation Specification makes it possible to specify:

- Spring beans that are made available to SCA as component services
- Spring beans that represent SCA properties
- Spring beans that represent SCA references

Three elements: `sca:service`, `sca:reference` and `sca:property`, can be used in a Spring application context configuration to identify a SCA service, a SCA reference or a SCA property, respectively.



**Figure 2-1: Example of EIS SCA prototype based on existing Spring application**

## 2.2 Java EE Technologies

Java Platform, Enterprise Edition 5 (Java EE 5) is used for SSN system implementation.

Key technologies in Java EE 5 include the following:

Presentationlayer

- Java Servlet
- JavaServer Faces
- Web application internationalization and localization

Enterprise JavaBeans (EJB) 3.0.

- Session beans
- Message-driven beans
- Timer Services

Platform services

- Transactions
- Resource connections
- Security
- Java Message Service (JMS)

## 2.3 Industry Standards

Web Services including SOAP and XML

## 2.4 Business Processes

The methodology used for gathering all business processes identified on the —Business Process Modelling Notation, also called BPMN (reference [R2]).

The primary goal of the BPMN effort is to provide a notation that is readily understandable by all business users, from the business analysts that create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and finally, to the business people who will manage and monitor those processes.

Thus, BPMN creates a standardized bridge for the gap between the business process design and process implementation.

BPMN defines a Business Process Diagram (BPD), which is based on a flowcharting technique tailored for creating graphical models of business process operations.

A Business Process Model, then, is a network of graphical objects, which are activities (i.e., work) and the flow controls that define their order of performance.

## 2.5 Open Source Frameworks

### 2.5.1 Spring framework

Spring Framework is a Java platform that provides comprehensive infrastructure support for developing Java applications. Spring facilitates the applications building from “plain old Java objects” (POJOs) and to apply enterprise services non-invasively to POJOs. This capability applies to the Java SE programming model and to full and partial Java EE.

Some of the Spring platform advantages are:

- Make a Java method execute in a transaction without having to deal with transaction APIs.
- Make a local Java method a remote procedure without having to deal with remote APIs.
- Make a local Java method a message handler without having to deal with JMS APIs.

Spring Framework version 3.0 shall be used; based on Java 5, and Java 6 is fully supported.

#### 2.5.1.1 Service layer: Spring Web Services

Spring Web Services is a product of the Spring framework focused on creating document-driven Web services. Spring Web Services aims to facilitate contract-first SOAP service development, allowing for the creation of flexible web services using one of the many ways to manipulate XML payloads.

Spring-WS supports multiple transport protocols. The most common is the HTTP transport, for which a custom servlet is supplied, but it is also possible to send messages over JMS, and even email.

The Spring-WS also supports XML API and XML marshalling and un-marshalling.

The advantages of using Spring-WS are :

- Spring based.
- Pluggability.
- Focus on SOAP.
- Use available implementations.
- Sensible defaults.
- Fully message based.

Spring-WS are being deployed as simple Web Modules. The Web Module defines a Dispatcher Servlet which is an alternative to the standard Spring-MVC Dispatcher Servlet with separate Adapters for the messages and the wsdl definitions. The Servlet detects automatically any wsdl definition defined in its application context. The wsdl is exposed under its bean name. The servlet also detects Endpoint



Adapters which are interfaces implemented for each endpoint type in order to handle separate SOAP requests.

The Server side of Spring-WS is designed around a central class that dispatches incoming XML messages to endpoints. The Spring-WS's MessageDispatcher is extremely flexible, allowing the use of any sort of class to an endpoint, as long as it can be configured in the spring IoC container.

EndpointAdapter provides the interface that must be implemented for each endpoint type to handle a message request. In Spring-WS, Endpoints are handling incoming XML messages. Message endpoints give access to the entire XML message including SOAP headers. The payload is simply the contents of the SOAP body. How incoming messages are routed to these endpoints is the responsibility of and EndpointMapping. The routing can be done based on different criteria. Spring-WS offers many out of the box implementations not only for the mapping criteria but also for the endpoint implementations.

Spring Web Services supports server-side JMS handling through the JMS functionality provided in the Spring framework. Spring Web Services provides the WebServiceMessageListener to plug in to a MessageContainer. This message listener requires a WebServiceMessageFactory to and MessageDispatcher to operate.

The Spring-WS includes also JMS transports. Besides the standard JMS configuration(connection factory and destination name to listen to ), we only have to define a WebServiceMessageListener and give it a reference to the message factory we are using and the message dispatcher.

#### **1.1.1.1 Platform services: Spring JMS**

The JMS API exposes two types of send methods, one that takes delivery mode, priority, and time-to-live as Quality of Service (QOS) parameters and one that takes no QOS parameters which uses default values. Since there are many send methods in JmsTemplate, the setting of the QOS parameters have been exposed as bean properties to avoid duplication in the number of send methods. Similarly, the timeout value for synchronous receive calls is set using the property setReceiveTimeout. Some JMS providers allow the setting of default QOS values administratively through the configuration of the ConnectionFactory. This has the effect that a call to MessageProducer's send method send(Destination destination, Message message) will use different QOS default values than those specified in the JMS specification. In order to provide consistent management of QOS values, the JmsTemplate must therefore be specifically enabled to use its own QOS values by setting the boolean property isExplicitQosEnabled to true.

## **2.5.2 Hibernate framework**

Hibernate is an object-relational mapping (ORM) library for the Java language, providing a framework for mapping an object-oriented domain model to a traditional relational database. Hibernate solves object-relational impedance mismatch problems by replacing direct persistence-related database accesses with high-level object handling functions.

Hibernate's primary feature is mapping from Java classes to database tables (and from Java data types to SQL data types). Hibernate provides the development of persistent classes following common Java idiom - including association, inheritance, polymorphism, composition and the Java collections framework. Hibernate also provides data query and retrieval facilities. Hibernate generates the SQL calls and relieves the developer from manual result set handling and object conversion, keeping the application portable to all supported SQL databases.

Hibernate provides a variety of the characteristics that SDO provides. For example, Hibernate provides the convenient static data APIs. Hibernate provides the optimistic concurrency, disconnected model. Hibernate framework could be extended to become SDO-capable data access services, which allows this framework to work within the SDO solution.

## **2.6 Application Server**

The WebLogic Server 12c shall be used to host SSN system, which provides the following functionality

- clustering feature provides increased scalability, availability and reliability
- WebLogic Server is compliant with Java Secure Socket Extension (JSSE). JSSE is a set of packages that support and implement the SSL and TLS v1 protocol.

WebLogic Server provides Secure Sockets Layer (SSL) support for encrypting data transmitted across WebLogic Server clients, as well as other servers.

WebLogic Server supports the RSA cipher suites listed in COTS related documentation.

## 2.7 Database

Oracle RDBMS Server 11g shall be used for SSN data storage; Real Application Clusters (RAC) technology provides fault tolerance, security, load balancing, and scalability.

## 2.8 Non-functional requirements

The non-functional requirements are addressed separately in terms of the architectural solution offered:

- Portability: The technical architecture solution is based on standard Java EE technologies. Any application server specific extensions/frameworks should be avoided where possible. Non portable parts of the SSN System, if applicable, should be isolated and documented.
- Scalability: The design of the system and selection of technologies support a scalable solution.  
  
In particular, some SSN services components may be deployed onto a separate host machine to address this requirement. However, considering the WebLogic Server Clustering feature, a homogenous SSN system deployment will provide increased performance due to the lack of remote calls between the SSN components.  
  
Additionally, the SSN System includes software frameworks/products that are known to be scalable.
- Reusability: A design based on components lends support for reusability. In addition, exposing these components as web services should further support this requirement.
- Modularity: The SSN System is delivered as software components.
- Maintainability: The components are loosely coupled therefore they should require less maintenance overhead.
- Availability: The technical solution allows for the SSN System to be deployed onto more than one application server to meet availability demands.
- Performance: The design of the SSN System and selection of software products/frameworks has been undertaken with performance in mind. As the SSN System may be deployed into different environments, one possible solution to addressing this requirement may be offered through the scalability capabilities of the application as it allows the workload to be distributed to more than one SSN application and therefore increasing available resources allocated to service user requests.




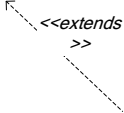
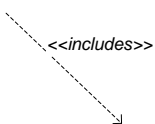
## 3 Overall System Architecture

### 3.1 Functional Architecture

For the purposes of the use case definition the following actors and external systems are identified:

- Ship data provider (human via web or system via XML/SOAP) represents the external systems that submit regulatory information to NSW.
- Authorities (human): requests for clearance are being routed to authorities based on the distribution rules defined by the national administrator. Authorities may have one or several functions (e.g. Port authority, Hazmat authority, Security authority, Waste authority, Border control authority, Customs authority, Health authority, Port State Control authority).
- The national administrator (human) is in charge of the management of user accounts, use profiles and configuration of the NSW.
- SSN Central (system) will provide the revamp PortPlus message exchange services and ShipCall requests with NSW.

For identifying the functionality the typical UML use cases notation is used:

Symbol	Description
	<b>Use Case</b> <ul style="list-style-type: none"> <li>▪ Represents a discrete unit of interaction between a user (human or machine) and the system.</li> <li>▪ Each Use Case has a description which describes the functionality that will be built in the proposed system. A Use Case may 'include' another Use Case's functionality or extend another Use Case with its own behaviour.</li> <li>▪ Use Cases are typically related to 'actors'.</li> </ul>
	<b>Actor</b> <ul style="list-style-type: none"> <li>▪ Human or machine entity that interacts with the system to perform meaningful work.</li> </ul>
	<b>Association</b> <ul style="list-style-type: none"> <li>▪ A relationship between two or more entities. Implies a connection of some type, for example one entity uses the services of another or one entity is connected to another over a network link.</li> </ul>
	<b>Extends Relationship</b> <ul style="list-style-type: none"> <li>▪ A relationship between two use cases in which one use case extends the behaviour of another.</li> </ul>
	<b>Includes Relationship</b> <ul style="list-style-type: none"> <li>▪ A relationship between two use cases in which one use case includes the behaviour.</li> </ul>

**Table 3-1: Use Cases Notation**

The following diagram provides an overview of the Use Cases.

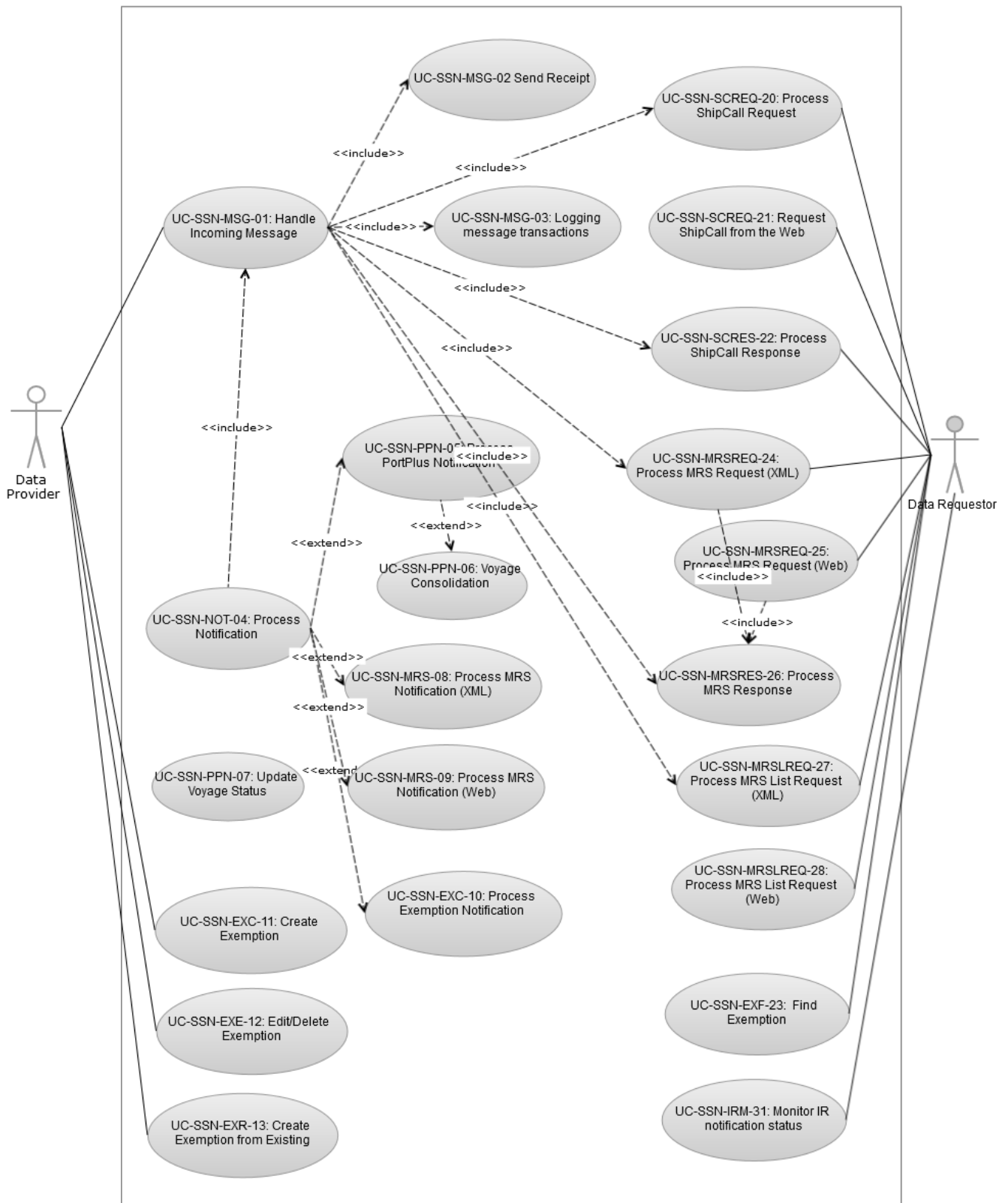


Figure 3-1SSN specific Use Case diagram

### 3.1.1 Handle Incoming Message

This system package includes the services required in order for the SSN Central to process messages via the XML/SOAP. This system package consists of the following use-case:

1. UC-SSN-MSG-01: Handle Incoming Message
2. UC-SSN-MSG-02: Send Receipt
3. UC-SSN-MSG-03: Logging message transactions

Use Case Req ID	<b>UC-SSN-MSG-01</b>	
Use Case Name	<b>Handle Incoming Message</b>	
Purpose	Covers the functionality related to the system's actions upon receiving a message either from Web or National Application.	
Subsystem	SSN Core	
Primary Actor(s)	SSN User / NCA Application / STIRES	
Precondition(s)	Actor is authorised for accessing to the system.	
Postcondition(s)	System has identified the message scope.	
Trigger(s)	Message Reception from Actor.	
Use Case Description	Primary Scenario	
Step 1	The system upon reception of the message checks its integrity and validates the message according to the XML Reference Guide.	
Step 2	After successful integrity check, the system identifies the scope and type of the received message (Notification/Alert/Information Request/Response).	
Step 3	After the type of message is identified, the system checks whether the Sender of the message is authorised to send such message types.	
Step 4	SSN time stamps (in UTC) the incoming message with the time of receipt.	
Step 5	Sends a receipt to the SSN User that sent the message.	
Alternative Use Case Description	Invalid Message Structure	
Step 1.1	The system identifies that the message is not correctly structured.	
Step 1.2	Logs the invalid message and sends a receipt to the SSN User synchronously.	
Alternative Use Case Description	Unidentified message type	
Step 2.1	The system cannot identify the received message as one of the predefined message types.	
Step 2.2	Go to 1.2	
Alternative Use Case Description	Not Authorised User	
Step 3.1	The system identifies that the sender is not authorised to send such	

	message types.
Step 3.2	Go to 1.2
Alternative Use Case Description	Not Unique MsRefId
Step 5.1	The system identifies that Message Reference Id of an XML message received from an NCA Application is not unique.
Step 5.2	Go to 1.2
Alternative Use Case Description	Tracking Service Request
Step 6.1	If there is a tracking service request associated to the specific message type, the system shall send the requested email to the subscribers.
Input(s)	XML Message / Information submitted from Web.
Output(s)	Identified scope of Message
Timer(s)	-
Business Process(es) Reference	-
Associated Use Case(s)	Included Case: UC-SSN-MSG-02 Send Receipt Included Case: UC-SSN-MSG-03: Logging message transactions
Special Requirements	When a message is sent through NCA Application, it shall be constructed as an XML and processed as such by the SSN Core. XML messages shall be uniquely identified by a unique message reference ID.

Use Case Req ID	<b>UC-SSN-MSG-02</b>	
Use Case Name	<b>Send Receipt</b>	
Purpose	The current use case describes the system's functionality related to the receipt sending for a received message.	
Subsystem	SSN Core	
Primary Actor(s)	SSN User/NCA Application	
Precondition(s)	A new message has been received from the system either through the SSN Web or XML Interface.	
Postcondition(s)	A receipt message is sent to the sender of a message.	
Trigger(s)	Validation results of a received message.	
Use Case Description	Primary Workflow	
Step 1	The system identifies the message type, if the message is formatted correctly as XML and if the sender is an NCA Application.	
Step 2	The system generates a receipt which acknowledges that the XML message is received and assigns the StatusCode ="OK" if the message was valid or StatuCode="InvalidFormat" if the message was invalid.	
Step 3	The system sends the receipt to the NCA Application.	

Input(s)	Validation Results.
Output(s)	Receipt message.
Timer(s)	-
Business Process(es) Reference	-
Associated Use Case(s)	-
Special Requirements	The form of the receipt message depends on the Interface used for the submission of the initial message (XML or Web).

Use Case Req ID	<b>UC-SSN-MSG-03</b>	
Use Case Name	<b>Logging message transactions</b>	
Purpose	The current use case describes the system functionality related to the logging of message transactions of SafeSeaNet events.	
Subsystem	Log Mechanism	
Primary Actor(s)	SSN Administrator	
Precondition(s)	System is operational and may send and accept messages.	
Postcondition(s)	Information related to the exchanged message is inserted in the system's log file.	
Trigger(s)	Any information exchange performed through the SafeSeaNet (Message Exchange, Receipts sent and received, Acknowledgments, Storage in database).	
Use Case Description	Primary Workflow	
Step 1	The system updates the log database with the exchanged data after each message related transaction of the system with the SSN users/NCA Applications. Data stored include: <ul style="list-style-type: none"> <li>• Message type (if XML complaint message)</li> <li>• Sender (if XML complaint message)</li> <li>• Timestamp</li> <li>• MsRefId of the message (if XML complaint message)</li> <li>• XML content</li> </ul>	
Step 2	The SSN Admin is provided with access to the systems' logged information through the Central Index database.	
Input(s)	Information related to message exchange.	
Output(s)	Updated log information.	
Timer(s)	-	
Business Process(es) Reference	-	
Associated Use Case(s)	-	
Special Requirements	The system keeps the log information in the Central Index database.	

### 3.1.2 Data Provide

This system package includes the services required in order for the SSN Central to process notification messages via the XML/SOAP and Web interfaces. More specifically, this system package consists of the following use-cases:

1. UC-SSN-NOT-04: Process Notification
2. UC-SSN-PPN-05: Process V3 PortPlus Notification
3. UC-SSN-PPN-06: Voyage Consolidation
4. UC-SSN-PPN-07: Update Voyage Status
5. UC-SSN-MRS-08: Process MRS Notification (XML)
6. UC-SSN-MRS-09: Process MRS Notification (Web)
7. UC-SSN-EXC-10: Process Exemption Notification
8. UC-SSN-EXC-11: Create Exemption
9. UC-SSN-EXE-12:Edit/Delete Exemption
10. UC-SSN-EXR-13:Create Exemption from Existing

Use Case Req ID	<b>UC-SSN-NOT-04</b>	
Use Case Name	<b>Process Notification</b>	
Purpose	Covers the functionality related to notification processing.	
Subsystem	SSN Core	
Primary Actor(s)	SSN User / NCA Application	
Precondition(s)	Handle Incoming Message UC has identified the incoming message as a valid Generic Notification.	
Postcondition(s)	System sends a receipt to the Data Provider about the acceptance of the notification and the transaction is logged.	
Trigger(s)	Notification Received	
Use Case Description	Primary Scenario	
Step 1	<p>The system identifies that the following information are contained in the notification:</p> <p>Data Provider;</p> <p>Vessel Identification (IMO, MMSI, Call Sign, Ship Name). The system identifies the vessel in the SSN database;</p> <p>The system also identifies that the received notification has one of the following types:</p> <ul style="list-style-type: none"> <li>• PortPlus notification;</li> <li>• Ship notification (AIS or MRS);</li> <li>• Incident notification;</li> <li>• Exemption notification.</li> </ul>	
Step 2	The system validates the notification applying the business rules for the specific Notification type (as defined in the XML Reference Guide	



	document) and also checks whether the Actor is Activated.
Step 3	The system persists the extracted information in the database and sends the Acknowledgement receipt to the data provider in order to indicate the successful reception and processing of the message details.
Alternative Use Case Description	Unidentified Vessel
Step 1.1	In case of a notification of type Ship, PortPlus, Alert for identified vessel, IR for identified vessel or Exemption the vessel information is missing both IMO and MMSI numbers.
Step 1.2	The system sends an error receipt to the data provider.
Alternative Use Case Description	Single Hull Tanker or Banned Vessel
Step 2.1	The system identifies that the received notification is about a Single Hull Tanker or a Banned Vessel.
Step 2.2	The notification is processed and linked to the vessel. A warning is produced to be appended in the SSN_Receipt message. The flow continues from step 3 or the Primary Workflow
Alternative Use Case Description	Validation Failure
Step 3.1	If the system fails to validate the received notification against the business rules for the identified notification type.
Step 3.2	Go to Step 1.2
Alternative Use Case Description	No Vessel in SSN DB
Step 4.1	If the system fails to identify the vessel in the SSN database, the system shall create a new entry in the database for the new vessel with status Temporary. The new vessel shall be associated to the received notification.
Alternative Use Case Description	The Vessel in SSN DB is updated
Step 5.1	If the system identifies the vessel's particulars reported in the notification are updated compared to the vessel in the SSN database, the system shall update the identified vessel's data by creating a new version of the vessel with status temporary and associated to the received notification
Alternative Use Case Description	Unidentified Port
Step 6.1	In case of Ship, PortPlus and Exemption notification if the system fails to identify thePorts ("Next Port of Call", "Port of Call", "Last Port", "Route.Port") in the SSN database the system shall check whether the LOCODE is correctly structured.  In case the LOCODE is valid the system shall store it in the database as temporary and accept the notification.  In case the LOCODE is invalid the system shall reject the notification.
Step 6.2	Go to Step 3 of the primary workflow.

Input(s)	New Notification
Output(s)	Notification passed to appropriate UC (PortPlus, Ship, Incident, Exemption)
Timer(s)	-
Business Process(es) Reference	-
Associated Use Case(s)	Included Case: UC-SSN-MSG-01: Handle Incoming Message Extended Case: UC-SSN-PPN-05: Process V3 PortPlus Notification, UC-SSN-MRS-08a: Process V3 MRS Notification, UC-SSN-MRS-08b: Process V2 MRS Notification.
Special Requirements	Submission of notification can be achieved either through XML or Web interface. MRS Notifications are only submitted through the web interface while AIS Notifications through the XML interface.

Use Case Req ID	<b>UC-SSN-PPN-05</b>
Use Case Name	<b>Process V3 PortPlus Notification</b>
Purpose	Covers the functionality related to V3 PortPlus Notification processing.
Subsystem	SSN Core
Primary Actor(s)	NCA Application
Precondition(s)	Process Notification UChas identified the incoming message as a valid PortPlus Notification.
Postcondition(s)	System persists Data in the SSN Database
Trigger(s)	Notification Received
Use Case Description	Valid New PortPlus Notification
Step 1	The system identifies that the Port Plus notification is a v3 PortPlus notification.
Step 2	The PortPlus UpdateStatus = N. The system checks that the ShipCallId is unique.
Step 3	The system also performs some additional authorization checks for the Actor submitting the request: <ol style="list-style-type: none"> <li>1. The Actor is known to SSN application.</li> <li>2. The Actor has been activated.</li> <li>3. The Actor validity period has not elapsed yet.</li> <li>4. The Actor has been granted the required permissions.</li> <li>5. The Actor is authorized to submit requests from XML interface.</li> </ol>
Step 4	The system validates the notification contents against the XMLRG business rules.
Step 5	The system sets the status of the PortPlus validation to Valid.

Step 6	The system persists the notification data in the DB as a new record. The notification data are stored as reported and then they are sent to the voyage consolidation process. The business rules defined in Annex A: Business Rules are applicable.
Step 7	The system Logs the message and processing details.
Alternative Use Case Description	Valid Update PortPlus Notification
Step 1.1	The PortPlus UpdateStatus = U.
Step 1.2	The system checks if the ShipCallId exists in the database and if yes it is unique for the MS of the UserId reporting the notification.  The voyage is identified and the status is not Canceled (perform step
Step 1.3	The system validates the notification contents against the XMLRG business rules.  The system checks if the ArrivalNotificationDetails and DepartureNotificationDetails are recorded in the corresponding voyage and if yes they are reported in the Update PortPlus Notification.
Step 1.4	The system sets the status of the PortPlus validation to Valid.
Step 1.5	The system persists the notification data in the DB as a new record and sends to the voyage consolidation process. The business rules defined in Annex A: Business Rules are applicable.
Step 1.6	The system Logs the message and processing details.
Alternative Use Case Description	Not Unique New PortPlus Notification
Step 2.1	The system identifies that for UpdateStatus = N the ShipCallId is not unique in the DB. This is done by querying the PORTPLUS_NOTIFICATIONS table by the reported ShipCallId and UpdateStatus='N'. If found then the ShipCallId is not unique.
Step 2.2	The system sets the status of the PortPlus validation to Invalid.
Alternative Use Case Description	ShipCallId not in DB for Update Notification
Step 3.1	The system identifies that for UpdateStatus = U the ShipCallId does not exist in the DataBase.
Step 3.2	The system checks that the ShipCallId does not exist in the database.
Step 3.3	The system validates the notification contents against the XMLRG business rules.
Step 3.4	The system sets the status of the PortPlus validation to Valid.
Step 3.5	The system produces a warning to be appended in the SSN_Receipt message that the corresponding PortPlus with UpdateStatus=N shall be sent.  The system produced an e-mail warning to the 24/7 NCA to request the NCA to send the original message as soon as possible.

Step 3.6	The system persists the notification data in the DB and sends to the voyage consolidation process. The business rules defined in Annex A: Business Rules are applicable.
Alternative Use Case Description	Cancel a PortPlus notification
Step 4.1	The system identifies that for UpdateStatus = U the ShipCallId exist in the DataBase. PortOfCall = ZZCAN.
Step 4.2	The system checks that the ShipCallId exist in the database.
	The system validates the notification contents against the XMLRG business rules. No ATA to PortOfcall is reported for the voyage.
Step 4.3	The system sets the status of the PortPlus validation to Valid.
Step 4.4	The system persists the notification data in the DB and sends to the voyage consolidation process. The business rules defined in Annex A: Business Rules are applicable.
Alternative Use Case Description	New PortPlus Notification - Invalid
Step 5.3	Validation of the submitted data fails against the XMLRG business rules.
Step 5.4	The system sets the status of the PortPlus validation to Valid. The error message will be communicated in the SSN_Receipt. Go to Step 5.6.
Step 5.5	Not executed.
Input(s)	New PortPlus Notification
Output(s)	Notification Information stored in the SSN database.
Timer(s)	-
Business Process(es) Reference	-
Associated Use Case(s)	Extended UC: UC-SSN-NOT-04 - Process Notification
Special Requirements	The new V3 PortPlus notification messages are only supported by the SSN System Interface (XML/SOAP). The V2 PortPlus notification messages continue to be supported by both the SSN System Interface (XML/SOAP) and the SSN Textual Interface.

Use Case Req ID	<b>UC-SSN-PPN-06</b>	
Use Case Name	<b>Voyage Consolidation</b>	
Purpose	Covers the functionality related to the voyage consolidation processing. The primary scope of the voyage calculation is to identify all the notifications transmitted to SSN-EIS that refer to the same ship call (defined in SafeSeaNet as voyage). PortPlus notifications sent for the	

	<p>same ShipCallID are parts of the same voyage. However, a voyage may include notifications sent by more than one data providers and therefore different ShipCallID. A voyage of the same vessel may be reported by the departure port (as NextPort) and by the arrival port (as PortOfCall). These two cases which are associated to the same voyage shall be correlated based on the vessel, port and ETA/ATA.</p> <p>The VoyageId is defined as [ShipCallId  '#'  Country alpha-2 ISO code].</p> <p>The PortPlus notification may be reported by a NCA application or by STIRES. In the later case the PortPlus notification reports calculated data, and the voyage to be created is identified as "detected". The process will consolidate PortPlus notification reported from NCA applications only with "non-detected" voyage, while the PortPlus notifications from STIRES will be consolidated only with "detected" voyages. A detected voyage may be co-related with a non-detected voyage if reported for the same ship and port of call and according to the rules defined in Annex A (Business Rules for calculated ATA, ATD and ETD).</p>
Subsystem	SSN Core
Primary Actor(s)	NCA Application/STIRES
Precondition(s)	Process PortPlus Notification UC has identified the incoming message as a valid PortPlus Notification.
Postcondition(s)	System persists Data in the SSN Database
Trigger(s)	PortPlus Notification Received
Use Case Description	Valid PortPlus Notification –reported from NCA Application
Step 1	The process checks in the data store if exists a voyage with VoyageId equal to the VoyageId of the notification.
Step 2a	<p>A voyage exists with the same VoyageId. Check if the new PortPlus notification has HazmatNonEUDeparture and/or CrewAndPaxNotificationOnArrival:</p> <ul style="list-style-type: none"> <li>• If yes, then update the voyage with the data from the new PortPlus.</li> <li>• If no, then update the voyage with the data from the new PortPlus. Then check for active Hazmat and/or CrewAndPax for the same ship, reported by a different data provider, with ETA in the future compared to the ETA reported in the notification, with ATA and ATD null. If found then assign the Hazmat and/or CrewAndPax to the new voyage created and delete Hazmat and/or CrewAndPax from the previous voyage.</li> </ul> <p>If the notification reports Security and/or Waste then if the existing voyage already has these data update with the corresponding new data; if the voyage has no such data then insert the new data.</p> <p>If the new PortPlus reports NextPort go to step 3 else go to step 5.</p>
Step 2b	<p>No voyage exists with the same VoyageId. Check for existing voyage that match the ship and PortOfCall and has neither Hazmat nor CrewAndPax.</p> <ul style="list-style-type: none"> <li>• If no existing voyage found and the new PortPlus has Hazmat</li> </ul>

	<p>and/or CrewAndPax then create a new voyage with the data reported in the new PortPlus notification. Insert also the Security and/or Waste reported in the new notification.</p> <ul style="list-style-type: none"> <li>• If no existing voyage found and the new PortPlus has no Hazmat and/or CrewAndPax then create a new voyage with the data reported in the new PortPlus notification. Insert also the Security and/or Waste reported in the new notification. Then check for active Hazmat and/or CrewAndPax for the same ship, reported by a different data provider, with ETA in the future compared to the ETA reported in the notification, with ATA and ATD null. If found then assign the Hazmat and/or CrewAndPax to the new voyage created and delete Hazmat and/or CrewAndPax from the previous voyage.</li> <li>• If an existing voyage is found then check the data in the PortPlus with the data in the existing voyage based on the business rules defined in Annex A: Business Rules. If a match is found then update the existing voyage found with the data from the new PortPlus. If no match then create a new voyage with the data from the new PortPlus. Insert also the Security and/or Waste reported in the new notification.</li> </ul> <p>If the new PortPlus has UpdateStatus = "U", then set the status of the new voyage to "On-hold".</p> <p>If the new PortPlus reports NextPort go to step 3 else go to step 5.</p>
Step 3	The valid PortPlus notification reports NextPort and ETA to NextPort.
Step 4a	<p>The notification reports HazmatEUDeparture and/or CrewAndPaxNotificationOnDeparture. Check for existing voyage that match the ship and NextPort in the notification with the PortOfCall in the voyage, was reported by a different data provider and has no Hazmat and/or CrewAndPax:</p> <ul style="list-style-type: none"> <li>• If no match then create a new voyage with the data from the new PortPlus for the NextPort including the DepartureNotificationDetails. Insert also the Hazmat and/or CrewAndPax reported in the new notification.</li> <li>• If an existing voyage is found then check the data in the PortPlus with the data in the existing voyage based on the business rules defined in Annex A: Business Rules. If a match is found then update the existing voyage found with the data from the new PortPlus for the NextPort including the DepartureNotificationDetails. If no match then create a new voyage with the data from the new PortPlus for the NextPort including the DepartureNotificationDetails. Insert also the Hazmat and/or CrewAndPax if reported in the new notification.</li> </ul>
Step 4b	<p>The notification reports neither HazmatEUDeparture nor CrewAndPaxNotificationOnDeparture. Check for existing voyage that match the ship, the ShipCallId reported the Hazmat and/or CrewAndPax and their providers.</p> <p>If a voyage is found then update the existing voyage with data from the new PortPlus for the NextPort. If the voyage has Hazmat and/or</p>

	CrewAndPaxthen delete the specific data. Do not update LastPort, ETDFromLastPort, ETAToPortOfCall and ETDFromPortOfCall.
Step 5	The system persists the notification data in the VOYAGES specific tables that store the consolidated ship call information reported in the related notifications. The voyage is identified as Non-Detected (reported from an NCA application).
Alternative Use Case Description	Valid detected PortPlus Notification –reported from STIRES
Step 1.1	The process check in the data store if exists a voyage based on the ShipCallId. The voyage must be a detected one meaning it has been reported from STIRES.
Step 1.2a	<p>A detected voyage exists with the same ShipCallId. Check if the new PortPlus notification matches a voyage reported by an NCA application based on the ship and PortOfCall.</p> <ul style="list-style-type: none"> <li>• If found then check the data in the new PortPlus with the data in the NCA application voyage based on the business rules defined in Annex A: Business Rules for calculated ATA, ATD and ETD. If found set the existing detected voyage VoyageId value to the Database_Id of the matching voyage reported from an NCA application.</li> <li>• If not match is found set the detected voyage VoyageId = Null.</li> </ul> <p>Update the existing detected voyage with the data from the new PortPlus notification.</p>
Step 1.2b	<p>No detected voyage exists with the same ShipCallId. Check if the new PortPlus notification matches a voyage reported by an NCA application based on the ship and PortOfCall.</p> <ul style="list-style-type: none"> <li>• If found then check the data in the new PortPlus with the data in the NCA application voyage based on the business rules defined in Annex A: Business Rules for calculated ATA, ATD and ETD. If found set the new detected voyage VoyageId value to the Database_Id of the matching voyage reported from an NCA application.</li> <li>• If not found set the new detected voyage VoyageId = Null.</li> </ul> <p>Insert a new detected voyage with the data from the new PortPlus notification.</p>
Step 1.3	The system persists the notification data in the VOYAGES specific tables that store the consolidated ship call information reported in the related notifications. The voyage is identified as Detected (reported from STIRES).
Alternative Use Case Description	Delete HazmatNonEuPort / HazmatEUPort / Security / Waste / Crew&Passengers
Step 2.1	<p>The valid PortPlus notification has at least one of the following parameters set to Y</p> <ul style="list-style-type: none"> <li>– DeleteHazmatNotificationInfoNonEUDepartures</li> <li>– DeleteHazmatNotificationInfoEUDepartures</li> <li>– DeleteWasteNotification</li> <li>– DeleteSecurityNotification</li> </ul>

	– DeleteCrewAndPaxOnDeparture
Step 2.2a	<p>A voyage exists with the same VoyageId.  Update the existing voyage with the data from the new PortPlus notification.  Delete the data related to the HazmatNonEuPort / HazmatEUPort / Security / Waste / Crew&amp;Passengers if the corresponding parameter is Y.  Go to step 2.3</p>
Step 2.2b	<p>No voyage exists with the same VoyageId.  Do nothing. Cannot delete data from a non-existing voyage.  Use case ends.</p>
Step 2.3	<p>The system persists the notification data in the VOYAGES specific tables that store the consolidated ship call information reported in the related notifications.</p>
Alternative Use Case Description	Cancel a PortPlus notification
Step 3.1	<p>The valid PortPlus notification has UpdateStatus = "U" and PortOfCall = "ZZCAN". The process check in the data store if exists a voyage based on the VoyageId.</p>
Step 3.2a	<p>A voyage exists with the same VoyageId.  Update the existing voyage with the data from the new PortPlus notification. Set the voyage status to Cancelled. The voyage will not longer be considered for a ShipCall request or consolidation of a new notification.  Go to step 3.3.</p>
Step 3.2b	<p>No voyage exists with the same VoyageId.  Do nothing. Cannot cancel a non-existing voyage.  Use case ends.</p>
Step 3.3	<p>The system persists the notification data in the VOYAGES specific tables that store the consolidated ship call information reported in the related notifications.</p>
Input(s)	Valid PortPlus Notification
Output(s)	<p>Notification Information are consolidated in the database with previously sent PortPlus notification data related with the same ship voyage. The consolidated ship call data are stored in the VOYAGES tables. Details specific to Hazmat, Security, Waste and Crew&amp;Passengers are stored in the VOYAGES child tables specific to the data.</p>
Timer(s)	-
Business Process(es) Reference	-
Associated Use Case(s)	Extended UC: UC-SSN-PPN-05: Process PortPlus Notification
Special Requirements	<ul style="list-style-type: none"> <li>• The business rules defined in Annex A: Business Rules are applicable.</li> <li>• Voyages with status "Dummy" or "Closed" are not considered in the</li> </ul>



	voyage consolidation.
--	-----------------------

Use Case Req ID	<b>UC-SSN-PPN-07</b>	
Use Case Name	<b>Update Voyage Status</b>	
Purpose	<p>The current use case describes the system functionality related to the update of the voyage status based on the reported data.</p> <p>The update voyage status will run at predefined intervals to:</p> <ul style="list-style-type: none"> <li>– set the voyage Status = “Closed” based on the reported ATD and ETA.</li> <li>– set the voyage Status = “Dummy” based on the reported ETA and ATA to a PortOfCall.</li> </ul>	
Subsystem	SSN database	
Primary Actor(s)	Oracle Job	
Precondition(s)	System is operational	
Postcondition(s)	The status of voyages that satisfy the conditions defined in the Purpose are updated.	
Trigger(s)	Oracle job scheduled to run at predefined intervals.	
Use Case Description	Primary Workflow	
Step 1	<p>The system updates the voyages with</p> <ul style="list-style-type: none"> <li>- ETAToPortOfCall in the past and with no ATAPortOfCall AND</li> <li>- If there is at least one voyage to another port with a later ATAPortOfCall compared to the voyage’s ETAToPortOfCall (reported by a MS or detected in STIRES).</li> </ul> <p>By setting the Status=“Dummy”.</p>	
Step 2	<p>The system updates the voyages with:</p> <ul style="list-style-type: none"> <li>- ATDPortOfCall before the configurable archival period or</li> <li>- ETAToPortOfCall before the configurable archival period, if no ATAPortOfCall nor ATDPortOfCall</li> </ul> <p>By setting the Status=“Closed”.</p>	
Input(s)	Current timestamp	
Output(s)	Voyage status update.	
Timer(s)	Daily. Note: this can be rescheduled to run on a hourly basis.	
Business Process(es) Reference	-	
Associated Use Case(s)	-	
Special Requirements	The systems considers two distinct configurable archival periods: one applied to ATDPortOfCall, and one applied to ETAToPortOfCall	

Use Case Req ID	<b>UC-SSN-MRS-08a</b>
Use Case Name	<b>Process V3 MRS Notification (XML)</b>
Purpose	Covers the functionality related to V3 MRS Notification processing via the system to system XML interface.
Subsystem	SSN Core
Primary Actor(s)	NCA Application
Precondition(s)	Process Notification UChas identified the incoming message as a valid MRS Notification.
Postcondition(s)	System persists Data in the SSN Database
Trigger(s)	Notification Received
Use Case Description	Valid New MRS Notification
Step 1	The system identifies that the MRS Notification message is a V3 MRSNotification.
Step 2	The system checks that the MSRefId is unique.
Step 3	<p>The system also performs some additional authorization checks for the Actor submitting the request:</p> <ul style="list-style-type: none"> <li>6. The Actor is known to SSN application.</li> <li>7. The Actor has been activated.</li> <li>8. The Actor validity period has not elapsed yet.</li> <li>9. The Actor has been granted the required permissions.</li> <li>10. The Actor is authorized to submit requests from XML interface.</li> </ul>
Step 4	<p>The system validates the notification contents against XMLRG business rules.</p> <p>Furthermore, the system will verify that the actor is submitting the Notification for a MRS that is managed by its country (e.g. Poland can submit GDANREP notifications but not WETREP notifications). Reference information regarding member states entitled to provide MRS information will be maintained by central SSN authority. [SSNV3_2].</p>
Step5	The system checks the contents of the identification of the Coastal Station (attribute "CSTIdentification"). If the content does not match any authority responsible for a MRS in the EMSA Central Organisation Database (COD), a warning message will be returned in the StatusMessage in the SSN_Receipt.xml (no rejection). The message will state the following: "WARNING: the Coastal Station (attribute CSTIdentification) is unknown to the system". [SSNV3_4].
Step 6	The system persists the notification data in the DB as a new record.
Step 7	The system logs the message and processing details.
Alternative Use case Description	Duplicate message; the actor has already sent a message with the same MSRefId
Ste 2.1	The system returns an SSN Receipt with StatusCode="InvalidFormat"
Alternative Use Case	Authorization fails

Description	
Step 3.1	The system returns an SSN_Receipt with StatusCode="AccessDenied"
Alternative Use Case Description	Validation against XMLRG business rules fails
Step 4.1	The system returns an SSN_Receipt with StatusCode="InvalidFormat"
Input(s)	New MRS Notification
Output(s)	Notification Information stored in the SSN database.
Timer(s)	-
Business Process(es) Reference	-
Associated Use Case(s)	Extended UC: UC-SSN-NOT-04 - Process Notification
Special Requirements	

Use Case Req ID	<b>UC-SSN-MRS-08b</b>
Use Case Name	<b>Process V2 MRS Notification (XML)</b>
Purpose	Covers the functionality related to V2 MRS Notification processing via the system to system XML interface.
Subsystem	SSN Core
Primary Actor(s)	NCA Application
Precondition(s)	Process Notification UChas identified the incoming message as a valid MRS Notification.
Postcondition(s)	System persists Data in the SSN Database
Trigger(s)	Notification Received
Use Case Description	Valid New MRS Notification
Step 1	The system identifies that the MRS Notification message is a V2 MRSNotification.
Step 2	The system checks that the MSRefId is unique.
Step 3	<p>The system also performs some additional Authorization checks for the Actor submitting the request:</p> <ol style="list-style-type: none"> <li>1. The Actor is known to SSN application.</li> <li>2. The Actor has been activated.</li> <li>3. The Actor validity period has not elapsed yet.</li> <li>4. The Actor has been granted the required permissions.</li> <li>5. The Actor is authorized to submit requests from XML interface.</li> </ol>
Step 4	The system validates the notification contents against V2 MRS Notification XMLRG business rules.
Step 5	The system persists the notification data in the V3 DB recording value "undefined" to <i>MRSIdentification</i> , "X" to <i>anyDG</i> and the "sentAt" date to

	<i>reportingDateTime.</i>
Step 6	The system logs the message and processing details.
Alternative Use case Description	Duplicate message; the actor has already sent a message with the same MSRefId
Ste 2.1	The system returns an SSN Receipt with StatusCode="InvalidFormat"
Alternative Use Case Description	Authorization fails
Step 3.1	The system returns an SSN_Receipt with StatusCode="AccessDenied"
Alternative Use Case Description	Validation against XMLRG business rules fails
Step 4.1	The system returns an SSN_Receipt with StatusCode="InvalidFormat"
Input(s)	New MRS Notification
Output(s)	Notification Information stored in the SSN V3 database.
Timer(s)	-
Business Process(es) Reference	-
Associated Use Case(s)	Extended UC: UC-SSN-NOT-04 - Process Notification
Special Requirements	

Use Case Req ID	<b>UC-SSN-MRS-09</b>
Use Case Name	<b>Process V3 MRS Notification (Web)</b>
Purpose	<p>Covers the functionality related to V3 MRS Notification processing via the Web interface.</p> <p>This communication channel is an alternative to the XML based interface with central SSN application.</p> <p>In that sense the information they should provide from the alternative browser-based web channel should conceptually include the union of information defined in ship notification (MS2SSN_Ship_Not) message and ship response (SSN2MS_Ship_Res) message. Hereafter, SSN central application will be able to respond to ship notification requests without further consultation of the data provider.</p>
Subsystem	SSN Core, SSN Send Notifications console
Primary Actor(s)	SSN User
Precondition(s)	The Actor has been authenticated in the system and is authorised to create MRS Notifications.
Postcondition(s)	System persists Data in the SSN Database
Trigger(s)	The Actor has selected to provide MRS Notification for a vessel.
Use Case Description	Create a new, valid MRS Notification

Step 1	The Actor navigates to MRS notification screen (Generic Notifications > MRS Notification) and searches for the vessel to send the MRS Notification for.	
Step 2	The system returns a list of vessels corresponding to the search criteria specified by the actor. The Actor selects the desired vessel and the system redirects to the screen where it prompts for MRS Notification details.	
Step 3	<p>The Actor fills in the information required to form a full MRS Notification and submits it. As explained above, this information merely includes non-duplicated information contained in both MS2SSN_Ship_Not and SSN2MS_Ship_Res messages.</p> <p>Since a Notification for an MRS can be submitted only from authorized users, the relevant list of MRS displayed in the input forms contains only the authorized MRS [SSNV3_2].</p>	
Step 4	The system successfully validates the notification contents against XMLRG business rules.	
Step 5	The system checks the contents of the identification of the Coastal Station (attribute "CSTIdentification"). If the content does not match any authority responsible for a MRS in the EMSA Central Organisation Database (COD), a warning message will be displayed in the success page returned to the actor. The message will state the following: "WARNING: the Coastal Station (attribute CSTIdentification) is unknown to the system". [SSNV3_4].	
Step 6	The system persists the notification data in the DB as a new record. The system will reuse the existing MrsNotification entity along with a new, optional, dependant entity MrsNotificationDetails for that purpose.	
Step 7	The system logs the message and processing details.	
Alternative Description	Use Case	The information provided for MRS Notification is not valid.
Step 2.1	The system returns to the input screen where all rule violations are presented and prompts the actor to correct them and re-submit the MrsNotification information.	
Step 1.2	The flow continues from Step 2.	
Input(s)	New MRS Notification	
Output(s)	Notification Information stored in the SSN database.	
Timer(s)	-	
Business Process(es) Reference	-	
Associated Use Case(s)	Extended UC: UC-SSN-NOT-04 - Process Notification	
Special Requirements		

Use Case Req ID	<b>UC-SSN-EXC-10</b>	
Use Case Name	<b>Process Exemption Notification</b>	
Purpose	Covers the functionality related to Exemption Notification processing.	
Subsystem	SSN Core	
Primary Actor(s)	NCA Application	
Precondition(s)	Process Notification UC has identified the incoming message as a valid Exemption Notification.	
Postcondition(s)	System persists Data in the SSN Database	
Trigger(s)	A notification XML message is received.	
Use Case Description	Valid New Exemption Notification	
Step 1	SSN is invoked to receive the incoming exemption notification or from an external application. UpdateStatus = N.	
Step 2	The system performs some authorization checks for the Actor submitting the request: <ol style="list-style-type: none"> <li>1. The Actor is known to SSN application.</li> <li>2. The Actor has been activated.</li> <li>3. The Actor validity period has not elapsed yet.</li> <li>4. The Actor has been granted the required permissions.</li> <li>5. The Actor is authorized to submit requests from XML interface.</li> </ol>	
Step 3	The system validates the reported data based on the XMLRG business rules.	
Step 4	The system persists the notification data in the DB as a new record. The notification data are stored as reported.	
Step 5	The system Logs the message and processing details.	
Alternative Use Case Description	InvalidNew Exemption Notification	
Step 1.6	Validation of the submitted data fails against the XMLRG business rules. The error message will be communicated in the SSN_Receipt.	
Alternative Use Case Description	UpdateExemption Notification	
Step 2.1	SSN is invoked to receive the incoming exemption notification or from an external application. UpdateStatus = U	
Step 2.2	The system performs some authorization checks for the Actor submitting the request: <ol style="list-style-type: none"> <li>1. The Actor is known to SSN application.</li> <li>2. The Actor has been activated.</li> <li>3. The Actor validity period has not elapsed yet.</li> <li>4. The Actor has been granted the required permissions.</li> </ol>	

	5. The Actor is authorized to submit requests from XML interface
Step 2.3	The system validates the reported data based on the XMLRG business rules.
Step 2.4	The systems identifies the exemption in the DB by ExemptionID.
Step 2.5	The system persists the notification data in the DB by updating the existing exemption with the updated data: all data of the existing exemption is replaced with the data from the notification (new data from the notification is added, existing data is replaced with the data from the notification, existing data which is not provided in the notification is deleted).
Step 2.6	The system Logs the message and processing details.
Alternative Use Case Description	DeleteExemption Notification
Step 3.1	SSN is invoked to receive the incoming exemption notification or from an external application. UpdateStatus = D
Step 3.2	The system performs some authorization checks for the Actor submitting the request: <ol style="list-style-type: none"> <li>1. The Actor is known to SSN application.</li> <li>2. The Actor has been activated.</li> <li>3. The Actor validity period has not elapsed yet.</li> <li>4. The Actor has been granted the required permissions.</li> <li>5. The Actor is authorized to submit requests from XML interface</li> </ol>
Step 3.3	The system validates the reported data based on the XMLRG business rules.
Step 3.4	The systems identifies the exemption in the DB by ExemptionID.
Step 3.5	The system deletes the existing exemption identified by ExemptionID.
Step 3.6	The system Logs the message and processing details.
Input(s)	Exemption information received as XML message for a vessel.
Output(s)	New exemption is submitted to the system and associated to a vessel.
Timer(s)	-
Business Process(es) Reference	-
Associated Use Case(s)	Extented UC: UC-SSN-NOT-04 - Process Notification
Special Requirements	-

Use Case Req ID	<b>UC-SSN-EXC-11</b>	
Use Case Name	<b>Create Exemption</b>	
Purpose	The current Use Case describes the SSN Users' interaction with the system upon creating a new exemption related to ship through the web.	

Subsystem	SSN Web Application
Primary Actor(s)	SSN User
Precondition(s)	The Actor has been authenticated in the system and is authorised to create exemptions.
Postcondition(s)	A new exemption is submitted in the system.
Trigger(s)	The Actor has selected to create an exemption for a vessel.
Use Case Description	Create an Exemption – Valid
Step 1	The Actor searches for a valid vessel to which the new exemption shall be associated. Search by IMO, MMSI, Call Sign and/or Ship Name is allowed.
Step 2	The system displays a list with the vessels that match the given criteria.
Step 3	The Actor selects a vessel from the list.
Step 4	<p>The system displays a form with fields for the new exemption.</p> <p>The provided fields are:</p> <p>Vessel Identification</p> <ul style="list-style-type: none"> <li>IMO Number</li> <li>MMSI Number</li> <li>Call Sign</li> <li>Ship Name</li> <li>Flag</li> </ul> <p>The fields that constitute the vessel information are filled with the data of the vessel that has been selected. Values can be updated.</p> <p>Exemption Details</p> <ul style="list-style-type: none"> <li>Exemption Type (Provided types are: "Pre-arrival", "Hazmat", "Waste" and "Security")</li> <li>Company Name</li> <li>Date From</li> <li>Date To</li> <li>Route, with list of: <ul style="list-style-type: none"> <li>- Port</li> </ul> </li> <li>Authority <ul style="list-style-type: none"> <li>- Country (based on the geographical restrictions applied to the user permission)</li> <li>- Authority Type</li> <li>- Authority Name</li> </ul> </li> <li>Contact 24/7 <ul style="list-style-type: none"> <li>- First Name</li> <li>- Last Name</li> <li>- Email</li> <li>- Phone</li> </ul> </li> </ul>



	- Fax  In addition the system generates a new Exemption ID (not visible on the form).
Step 5	The Actor keys-in exemption information to the provided fields and submits.
Step 6	The system validates the submitted information.
Step 7	The system persists the notification data in the DB. The notification data are stored as reported.
Step 8	The system Logs the message and processing details and informs the user about the result.
Alternative Use Case Description	Create an Exemption – Invalid
Step 1.6	Validation of the submitted information fails. The system displays the failure reason(s) to the Actor.
Step 1.7	Return to Primary Workflow – Step 2.
Alternative Use Case Description	Create an Exemption for a non-existing vessel
Step 2.2	No existing vessel matches the criteria.
Step 2.3	The Actor selects to create a new vessel (with status temporary)
Step 2.4	The system displays a form with fields for the new exemption (Vessel Identification and Exemption Details)
Step 2.5	The Actor keys-in exemption information to the provided fields and submits.
Step 2.6	The system checks if the data provided for the vessel and as long as they are technically correct, a new temporary vessel will be created and associated with the new exemption.
Alternative Use Case Description	Create an Exemption with invalid vessel input
Step 2.5	The Actor specifies invalid vessel.
Step 2.6	The system checks if the data provided for the vessel and prompts the user to enter valid vessel data.
Input(s)	Exemption information for a vessel.
Output(s)	New exemption is submitted to the system and associated to a vessel.
Timer(s)	-
Business Process(es) Reference	-
Associated Use Case(s)	-
Special Requirements	The actor can create an exemption associated to his/her own country (field "Authority - Country" in the exemption). This is based on the geographical restrictions applied to the permission EXEMPTIONS_NOTIFIER.

	<p>Validation rules are based on the Exemption Notification definition in [R4]</p> <p>A new Exemption ID is generated based on a dedicated database sequence that generates unique numbers. The Exemption_ID is the concatenation of the NSW Country ISO alpha-2 code and the sequence number.</p>
--	--

Use Case Req ID	<b>UC-SSN-EXE-12</b>	
Use Case Name	<b>Edit/Delete Exemption</b>	
Purpose	The current Use Case describes the SSN Users' interaction with the system upon editing-deleting an exemption related to ship through the web.	
Subsystem	SSN Web Application	
Primary Actor(s)	SSN User	
Precondition(s)	<p>The Actor has been authenticated in the system and is authorised to modify exemptions.</p> <p>The exemption is defined for a given vessel.</p>	
Postcondition(s)	An exemption is deleted or its data are modified.	
Trigger(s)	The Actor has selected to modify an exemption related to a vessel.	
Use Case Description	Primary Workflow	
Step 1	The Actor searches for an exception by entering Vessel and type of Exemption as criteria.	
Step 2	The system displays a list with exemptions that match the criteria and the user is allowed to edit.	
Step 3	The Actor selects an exemption from the list.	
Step 4	The system displays a form with fields filled in with the exemption data.	
Step 5	<p>The user modifies the exemption data (listed below) and submits.</p> <p>Exemption Details</p> <p>Exemption Type (Provided types are: "Pre-arrival", "Hazmat", "Waste" and "Security")</p> <p>Company Name</p> <p>Date From</p> <p>Date To</p> <p>Route, with list of:</p> <ul style="list-style-type: none"> <li>- Port</li> </ul> <p>Authority</p> <ul style="list-style-type: none"> <li>- Authority Type</li> <li>- Authority Name</li> </ul> <p>Contact 24/7</p>	

	<ul style="list-style-type: none"> <li>- First Name</li> <li>- Last Name</li> <li>- Email</li> <li>- Phone</li> <li>- Fax</li> </ul>
Step 6	The system validates the submitted data and displays the pertinent message to the user.
Alternative Use Case Description	Edit an Exemption - Invalid
Step 1.6	Validation of the submitted information fails. The system displays the failure reason(s) to the Actor.
Step 1.7	Return to Primary Workflow – Step 2.
Alternative Use Case Description	Delete an Exemption
Step 2.5	The actor selects to delete the selected exemption.
Step 2.6	The system deletes the exemption data.
Input(s)	Exemption information for a vessel.
Output(s)	An exemption's data are modified or deleted.
Timer(s)	-
Business Process(es) Reference	-
Associated Use Case(s)	-
Special Requirements	The actor can edit/delete an exemption associated to his/her own country (field "Authority - Country" in the exemption). This is based on the geographical restrictions applied to the permission EXEMPTIONS_NOTIFIER.

Use Case Req ID	<b>UC-SSN-EXR-13</b>	
Use Case Name	<b>Create Exemption from Existing</b>	
Purpose	The current Use Case describes the SSN Users' interaction with the system upon creating a new exemption related to ship through the web by using the data of an exemption that already exists.	
Subsystem	SSN Web Application	
Primary Actor(s)	SSN User	
Precondition(s)	The Actor has been authenticated in the system and is authorised to	

	create exemptions. Some exemptions already exists in the system.
Postcondition(s)	A new exemption is submitted in the system.
Trigger(s)	The Actor has selected to create an exemption for a vessel using the data of an existing exemption.
Use Case Description	Primary Workflow
Step 1	The Actor navigates to the page where he can create a new exemption from an existing one.
Step 2	The Actor searches for an exception by entering Vessel and type of Exemption as criteria.
Step 3	The system displays a form with fields for the new exemption populated with the values of the exemption that has been selected from the Actor.
Step 4	<p>The Actor modifies the following details:</p> <p>Exemption Details</p> <p>Exemption Type (Provided types are: "Pre-arrival", "Hazmat", "Waste" and "Security")</p> <p>Company Name</p> <p>Date From</p> <p>Date To</p> <p>Route, with list of:</p> <ul style="list-style-type: none"> <li>- Port</li> </ul> <p>Authority</p> <ul style="list-style-type: none"> <li>- Authority Type</li> <li>- Authority Name</li> </ul> <p>Contact 24/7</p> <ul style="list-style-type: none"> <li>- First Name</li> <li>- Last Name</li> <li>- Email</li> <li>- Phone</li> <li>- Fax</li> </ul>
Step 5	The system validates the submitted information.
Step 6	The system persists the notification data in the DB. The notification data are stored as reported.
Step 7	The system Logs the message and processing details and informs the user about the result.
Alternative Use Case Description	Create a new Exemption from an existing one - Invalid
Step 1.9	Validation of the submitted information fails. The system displays the failure reason(s) to the Actor.
Output(s)	New exemption is submitted to the system and associated to a vessel.
Timer(s)	-
Business Process(es) Reference	-

Associated Use Case(s)	-
Special Requirements	The actor can create an exemption associated to his/her own country (field "Authority - Country" in the exemption). This is based on the geographical restrictions applied to the permission EXEMPTIONS_NOTIFIER.

### 3.1.3 Data Request

This system package includes the services required in order for the SSN Central to process requests for details and response via the XML/SOAP and Web interfaces. More specifically, this system package consists of the following use-cases:

1. UC-SSN-SCREQ-20: Process ShipCall Request
2. UC-SSN-SCREQ-21: Request ShipCall from the Web
3. UC-SSN-SCRES-22: Process ShipCall Response
4. UC-SSN-EXF-23: Find Exemption
5. UC-SSN-MRSREQ-24: Process MRS Request (XML)
6. UC-SSN-MRSREQ-25: Request MRS Request(Web)
7. UC-SSN-MRSRES-26: Process MRS Response
8. UC-SSN-MRSLREQ-27: Process MRS List Request (XML)
9. UC-SSN-MRSLREQ-28: Process MRS List Request (Web)

Use Case Req ID	<b>UC-SSN-SCREQ-20</b>	
Use Case Name	<b>Process ShipCall Request</b>	
Purpose	Covers the functionality related to the systems' actions upon receiving a request for information from either the SSN GI Interface in v3 or a National Application in v2 or v3.	
Subsystem	SSN Core	
Primary Actor(s)	SSN User / NCA Application / SSN GI	
Precondition(s)	A v2 or v3 request for information has been received by the system.	
Postcondition(s)	The system has processed the request.	
Trigger(s)	Request message received	
Use Case Description	Primary Workflow	
Step 1	The SSN GI or an NCA application has sent a valid request for information to the System.	
Step 2	The system performs some authorization checks for the Actor submitting the request: <ol style="list-style-type: none"> <li>1. The Actor is known to SSN application.</li> <li>2. The Actor has been activated.</li> </ol>	

	<p>3. The Actor validity period has not elapsed yet.</p> <p>4. The Actor has been granted the required permissions.</p> <p>5. The Actor is authorized to submit requests from XML interface</p>
Step 3	<p>The system identifies access rights restrictions on ShipCall requests, based on the following:</p> <ul style="list-style-type: none"> <li>On the SHIPCALL_REQUESTOR task and geographical restrictions based on the PortOfCall.</li> <li>On the HAZMAT_REQUESTOR task if HazmatDetails are requested.</li> <li>On the SECURITY_REQUESTOR task if SecurityDetails are requested.</li> <li>On the WASTE_REQUESTOR task if WasteDetails are requested</li> <li>On the EXEMPTION_REQUESTOR task if Exemption information exists</li> <li>On the CREWPAX_REQUESTOR task if Crew and Passengers Details are requested.</li> </ul>
Step 4	<p>The System identifies one of the following requests (queries):</p> <p><i>ExpectedCallOfSelectedShip:</i> SSN protocol version: 2 and 3 Mandatory parameters: IMONumber or MMSINumber; Optional parameters: StartDateTime, GetHazmat, GetWaste, GetSecurity, GetCrewPax. Business rules applicable [R4] Table 4– Description of queries supported by the MS2SSN_ShipCall_Req message. Result: Details of a Voyage</p> <ul style="list-style-type: none"> <li>Without ATAPortOfCall, and</li> <li>With ETAToPortOfCall after and closest to StartDateTime.</li> <li>Hazmat/Security/Waste/Crew&amp;Pax information in results is as reported before departure from the port.</li> </ul> <p><i>MostRecentArrivalOfSelectedShip:</i> SSN protocol version: 2 and 3 Mandatory parameters: IMONumber or MMSINumber; Optional parameters: StartDateTime, GetHazmat, GetWaste, GetSecurity, GetCrewPax. Business rules applicable [R4] Table 4– Description of queries supported by the MS2SSN_ShipCall_Req message. Result: a Voyage</p> <ul style="list-style-type: none"> <li>With ATAPortOfCall before and closest to StartDateTime, and</li> <li>With ATDPortOfCall, if available, after StartDateTime.</li> <li>Hazmat/Security/Waste/Crew&amp;Pax information in results is as reported before departure from the port.</li> </ul> <p><i>MostRecentDepartureOfSelectedShip:</i> SSN protocol version: 2 and 3 Mandatory parameters: IMONumber or MMSINumber; Optional parameters: StartDateTime, GetHazmat, GetWaste, GetSecurity,</p>

GetCrewPax. Business rules applicable [R4] Table 4– Description of queries supported by the MS2SSN\_ShipCall\_Req message.

Result: a Voyage

- Central SSN will provide the ship call with ATDPortOfCall before and closest to StartDateTime.
- Hazmat/Security/Waste/Crew&Pax information in results is as reported before departure from the port.

*RecentAndCurrentShipCallsOfSelectedShip:*

SSN protocol version: 2 and 3

Mandatory parameters: IMONumber or MMSINumber; Optional parameters: StartDateTime, EndDateTime, NumberOfCalls. Business rules applicable [R4] Table 4– Description of queries supported by the MS2SSN\_ShipCall\_Req message.

Result: List of Voyages

- With ATAPortOfCall within the time period defined by StartDateTime and EndDateTime.
- If no time period is defined, Central SSN will provide the list of latest [NumberOfCalls] consolidated PortPlus messages with ATAPortOfCall before SentAt.
- Only summary data are provided. No request for additional details from the data provider.

*ExpectedShipCallsAtEUPort:*

SSN protocol version: 2 and 3

Mandatory parameters: PortOfCall; Optional parameters: StartDateTime, EndDateTime, NumberOfCalls. Business rules applicable [R4] Table 4– Description of queries supported by the MS2SSN\_ShipCall\_Req message.

Result: List of Voyages

- With ETAToPortOfCall within the time period defined by StartDateTime and EndDateTime, and
- Without ATAPortOfCall.
- If no time period is defined, Central SSN will provide the list of [NumberOfCalls] correlated voyages:
- With ETAToPortOfCall after SentAt, and
- Without ATAPortOfCall.
- Only summary data are provided. No request for additional details from the data provider.

*CurrentShipCallsAtEUPort:*

Mandatory parameters: PortOfCall; Optional parameters: StartDateTime. Business rules applicable [R4] Table 4– Description of queries supported by the MS2SSN\_ShipCall\_Req message.

Result: List of Voyages

- With PortOfCall = defined PortOfCall, and

- With ATAPortOfCall after StartDateTime, and
- Without ATDPortOfCall.
- Only summary data are provided. No request for additional details from the data provider.

*CompletedShipCallsAtEUPort:*

SSN protocol version: 2 and 3

Mandatory parameters: PortOfCall; Optional parameters: StartDateTime. Business rules applicable [R4] Table 4– Description of queries supported by the MS2SSN\_ShipCall\_Req message.

Result: List of Voyages

- With PortOfCall = defined PortOfCall, and
- With ATDPortOfCall after StartDateTime.
- Only summary data are provided. No request for additional details from the data provider.

*LatestRegisteredShipCallDataOfSelectedShip*

SSN protocol version: 2

Mandatory parameters: IMONumber or MMSINumber; StartDateTime and EndDateTime. Business rules applicable [R4] Table 4– Description of queries supported by the MS2SSN\_ShipCall\_Req message.

Result: Details of a Voyage.

- With the maximum sent at time less than StartDateTime.
- Hazmat information in results is as reported before departure from the port.

*ShipCallsRegisteredBySSNYesterday*

SSN protocol version: 2

Mandatory parameters: StartDateTime. Business rules applicable [R4] Table 4– Description of queries supported by the MS2SSN\_ShipCall\_Req message.

Result: List of Voyages which were registered or updated within the specified time period.

- Only summary data are provided. No request for additional details from the data provider.

*LatestCallUpdates:*

SSN protocol version: 2 and 3

Mandatory parameters: StartDateTime, EndDateTime. Business rules applicable [R4] Table 4– Description of queries supported by the MS2SSN\_ShipCall\_Req message.

Result: List of Voyages which were registered or updated within the specified time period.

- Only summary data are provided. No request for additional details from the data provider.



*LatestCallInfoAtSpecificPort*

SSN protocol version: 2

Mandatory parameters: StartDateTime, EndDateTime, PortOfCall. Business rules applicable [R4] Table 4– Description of queries supported by the MS2SSN\_ShipCall\_Req message.

Result: List of Voyages which were registered or updated within the specified time period.

- With PortOfCall = defined PortOfCall.
- Only summary data are provided. No request for additional details from the data provider.

*ListExpectedCallsOfSelectedShip:*

SSN protocol version: 2 and 3

Mandatory parameters: IMONumber or MMSINumber; Optional parameters: StartDateTime. Business rules applicable [R4] Table 4– Description of queries supported by the MS2SSN\_ShipCall\_Req message.

Result: List of Voyages

- Without ArrivalDetails element, and
- With ETAToPortOfCall after StartDateTime, and
- Without ATAPortOfCall.
- Only summary data are provided. No request for additional details from the data provider.

*SelectedShipCall:*

SSN protocol version: 3

Mandatory parameters: ShipCallID; Optional parameters: GetHazmat, GetWaste, GetSecurity, GetCrewPax. Business rules applicable [R4] Table 4– Description of queries supported by the MS2SSN\_ShipCall\_Req message.

Result: a Voyage

- With the specified ShipCallID.
- If the voyage does not have an ATD, the system will provide the Hazmat/Crew&Pax information in results as reported before departure from the port.
- If the voyage has an ATD, the system will provide the Hazmat/Crew&Pax information of the voyage towards next port.
- Security/Waste information in results as reported before departure from the port.

*GetActiveHazmatForSelectedShip:*

SSN protocol version: 2 and 3

Mandatory parameters: IMONumber or MMSINumber; Optional parameters: StartDateTime. Business rules applicable [R4] Table 4– Description of queries supported by the MS2SSN\_ShipCall\_Req message.

	<p>Result: a Voyage</p> <ul style="list-style-type: none"> <li>– that is the most relevant and associated hazmat details which are active at StartDateTime.</li> <li>– Information may come from different PortPlus messages (different values of ShipCallID).</li> <li>– Hazmat information in results.</li> </ul> <p><i>GetActiveSecurityForSelectedShip:</i></p> <p>SSN protocol version: 3</p> <p>Mandatory parameters: IMONumber or MMSINumber; Optional parameters: StartDateTime. Business rules applicable [R4] Table 4– Description of queries supported by the MS2SSN_ShipCall_Req message.</p> <p>Result: the most relevant Voyage and associated security details.</p> <ul style="list-style-type: none"> <li>– Without ATDPortOfCall and with closest ETAToPortOfCall or ATAPortOfCall to StartDateTime</li> <li>– Security information in results.</li> </ul> <p><i>GetActiveWasteForSelectedShip:</i></p> <p>SSN protocol version : 3</p> <p>Mandatory parameters: IMONumber or MMSINumber; Optional parameters: StartDateTime. Business rules applicable [R4] Table 4– Description of queries supported by the MS2SSN_ShipCall_Req message.</p> <p>Result: the most relevant Voyage and associated waste details.</p> <ul style="list-style-type: none"> <li>– Without ATDPortOfCall and</li> <li>– With closest ETAToPortOfCall or ATAPortOfCall to StartDateTime</li> <li>– Waste information in results. Details in case requested and are provided by the data provider. Summary in case requested or details are not provided by the data provider.</li> </ul> <p><i>GetActiveCrewPaxForSelectedShip:</i></p> <p>SSN protocol version: 2 and 3</p> <p>Mandatory parameters: IMONumber or MMSINumber; Optional parameters: StartDateTime. Business rules applicable [R4] Table 4– Description of queries supported by the MS2SSN_ShipCall_Req message.</p> <p>Result: the most relevant Voyage and associated waste details.</p> <ul style="list-style-type: none"> <li>– Without ATDPortOfCall and</li> <li>– With closest ETAToPortOfCall or ATAPortOfCall to StartDateTime</li> <li>– Crew&amp;Pax information in results.</li> </ul>
Step 5a	<p>In case of a v3 request, the system verifies that the search criteria are correct based on the XMLRG business rules for v3 request.</p> <p>The system searches the SSN database for a relevant voyage(s) based on the provided request criteria. The business rules for voyage retrieval defined in Annex A: Business Rules are also applicable.</p> <p>In case of a request for Hazmat/Waste/Security/Crew&amp;Passengers summary/details go to step 6a.</p>

	Otherwise go to step 6c.
Step 5b	<p>In case of a v2 request, the system verifies that the search criteria are correct based on the XMLRG business rules for v2 request.</p> <p>The system searches the SSN database for a relevant voyage(s) based on the provided request criteria. The business rules for voyage retrieval defined in Annex A: Business Rules are also applicable.</p> <p>In case of a request for Hazmat summary/details go to step 6b.</p> <p>Otherwise go to step 6c.</p>
Step 6a	<p>In case of a v3 request, the system has identified a voyage which complies with the search criteria for Hazmat/Waste/Security/Crew&amp;Passengers summary/details and the requestor's access rights and geographical restrictions (applied to the "PortOfCall" of the voyage: if is within the geo restriction, then the voyage is provided).</p> <p>The system fetches the voyage data from the data store. Go to step 7a.</p>
Step 6b	<p>In case of a v2 request, the system has identified a voyage which complies with the search criteria for Hazmat summary/details and the requestor's access rights and geographical restrictions (applied to the "PortOfCall" of the voyage: if is within the geo restriction, then the voyage is provided).</p> <p>The system fetches the voyage data from the data store. Go to step 7b.</p>
Step 6c	<p>The system has identified a list of voyage(s) which complies with the search criteria and the requestor's access rights and geographical restrictions (applied to the "PortOfCall" of each voyage: if is within the geo restriction, then the voyage is provided). The system fetches the voyage data from the data store. Go to step 10.</p>
Step 7a	<p>In case of a v3 request for Hazmat/Waste/Security/Crew&amp;Passengers details, the system identifies that the requested detailed information in XML must be acquired from an NCA Application (data provider). The system issues a ShipCall Request to the NCA application for the details.</p> <p>In case the details were reported in v3 PortPlus notification the system will issue a v3 SSN2MS_ShipCall_Req.</p> <p>In case the details are hazmat and were reported in v2 PortPlus notification the system will issue a v2 SSN2MS_ShipCall_Req.</p> <p>Go to step 8a.</p>
Step 7b	<p>In case of a v2 request for hazmat details, the system identifies that the details were reported in v2 PortPlus notification:</p> <ul style="list-style-type: none"> <li>– the requested detailed information are in UrlDetails. The system fetches the UrlDetails from the data store</li> <li>– the requested detailed information are in ContactDetails. The system fetches the ContactDetails from the data store.</li> <li>– the requested detailed information in XML must be acquired from an NCA Application (data provider). The system issues a v2 ShipCall Request to the NCA application for the details.</li> </ul> <p>In case the hazmat details were reported in v3 PortPlus notification the system will not issue a SSN2MS_ShipCall_Req for additional details. Instead the system will use the voyage data reported in the v3 PortPlus notification data (see step 6b) to use in the v2 response.</p> <p>Go to step 8b.</p>
Step 8a	<p>The system receives within the timeout interval the response from the NCA application. It parses the QueryResults part of the response</p>

	<p>message.</p> <p>In case of a v2 MS2SSN_ShipCall_Res, the contents are mapped to the contents of the v3 SSN2MS_ShipCall_Res message.</p> <p>As a general rule regarding the SSN2MS_ShipCall_Res message:</p> <ul style="list-style-type: none"> <li>• The following elements are excluded (independent of the GetDetails request) from the message: "PurposeOfCall", "VesselDetails", "WasteConfirmation", "SecurityConfirmation", "CrewAndPaxConfirmation", "Exemptions".</li> <li>• Under element HazmatSummary: <ul style="list-style-type: none"> <li>◦ If DGClassification = "BC" will be renamed to DGClassification = "IMSBC"</li> <li>◦ If DGClassification = "INF", then the element "DG" will be ignored</li> </ul> </li> <li>• Element "Source" in "HazmatDetails" indicates the "From" and "SentAt" of the MS2SSN_ShipCall_Res.</li> <li>• Only one V3 "Consignement" element is created. It has no attributes "TransportDocumentID", "PortOfLoading", "PortOfDischarge".</li> <li>• Each element "DPG" in "PlacementOfGoodsInContainer" of MS2SSN_ShipCall_Res is translated to element "DPGItem" with "TransportEquipmentUnit" element.</li> <li>• Attribute "LocationOnBoardGoods" in element "PlacementOfGoodsInContainer" of MS2SSN_ShipCall_Res is copied to attribute "LocationOnBoard" in element "TransportEquipmentUnit" of each "DPGItem" element.</li> <li>• Each element "DPG" in "PlacementOfGoods" of MS2SSN_ShipCall_Res is translated to element "DPGItem" with "NonTransportEquipmentUnit" element.</li> <li>• Attribute "LocationOnBoardGoods" in element "PlacementOfGoods" of MS2SSN_ShipCall_Res is copied to V3 attribute "LocationOnBoard" in element "NonTransportEquipmentUnit" of each "DPGItem" element.</li> <li>• As regards "DPGItem" elements: <ul style="list-style-type: none"> <li>◦ A dummy value = "X" will be given to HazmatDetails&gt; CargoInformation&gt; Consignment&gt; DPGItem&gt; DGClassification</li> <li>◦ Have no attributes "PackingGroup", "FlashPoint", "MarpolCode", "PackageType", "TotalNrOfPackages", "AdditionalInformation", "NoOfPackages".</li> <li>◦ Attribute "TechnicalName" of MS2SSN_ShipCall_Res is translated "TextualReference".</li> <li>◦ Have no element "EmS", "SubsidiaryRisks".</li> </ul> </li> </ul> <p>Go to step 9a.</p>
Step 8b	<p>The system receives within the timeout interval the response from the NCA application. It parses the QueryResults part of the response message.</p>

	Go to step 9b.
Step 9a	<p>In case of v3 request, if the requestor has the access rights to requesting exemption data, the system searches for exemptions in the SSN database which are relevant.</p> <p>An exemption is considered as relevant for a ship call if:</p> <ul style="list-style-type: none"> <li>• It applies to the same ship, and</li> <li>• Its route includes the Port Of Call, and</li> <li>• If the details are regarding Non-EU-Departure hazmat, or waste, or security, its validity period covers the ATAPortOfCall, or the ETAToPortOfCall (if there is no ATAPortOfCall), and</li> <li>• If the details are regarding EU-Departure Hazmat, its validity period covers the ATDPortOfCall, or the ETDFromPortOfCall (if there is no ATDPortOfCall).</li> </ul> <p>If one or more exemption are found, and, the details are fetched to be included in the response message.</p>
Step 9b	In case of v2 request, any relevant pre-arrival exemption for the requested ship and PortOfCall is ignored. The response is based on the relevant ship call data.
Step 10	<p>In case of a v2 request the system constructs a v2 response message.</p> <p>In case of a v3 request the system constructs a v3 response message.</p>
Step 11	The system sends the response to the data requestor.
Alternative Use Case Description	Invalid Request
Step 1.1	The ShipCall request criteria are invalid
Step 1.3	The system generates a receipt indicating the error and StatusCode = "InvalidFormat" and a StatusMessage indicating that the search criteria are incorrect.
Alternative Use Case Description	No relevant data found.
Step 2.1	The system identifies that no voyage satisfies the request criteria. The system generates and sends a negative response to the data requestor.
Step 2.3	The system generates a SSN2MS_ShipCall_Res with StatusCode = "NotFound".
Alternative Use Case Description	Exceeded Response Time/ Negative receipt
Step 3.1	If the response time (as defined in the request – see Special requirements) has been exceeded, or if a negative receipt has been received from the data provider's NCA Application, the system sends a Response message to the data requester with the summary that it holds in the database (notification details).
Alternative Use Case Description	Access Control Check fails
Step 4.1	<p>If the Access Control fails to apply the restrictions of Step 2 and 3,</p> <p>The system identifies that the user is not granted the permission to</p>

	submit the request (is not granted the relative TASK; see step 3) or the geographical access rights restrictions on ShipCall requests does not permit submitting a request for the PortOfCall indicated in the request criteria.
Step 4.2	The system generates a receipt indicating the error and StatusCode = "AccessDenied".
Input(s)	Request specific data.
Output(s)	Notification Details are sent to the Data Requester (SSN User or NCA Application). The system shall respond to a requesting NCA Application in XML format.  For Cargo Manifest the response message defines the URL where the document can be downloaded from or the contact details of the authority/person to acquire the details from.
Timer(s)	-
Business Process(es) Reference	Response Timer
Associated Use Case(s)	Included UC: UC-SSN-MSG-01: Handle Incoming Message
Special Requirements	<ol style="list-style-type: none"> <li>1. Once a request has been received, the system initiates a response timer. If the time interval to be applied is not defined in the request message the system sets the timer with a predefined value.</li> <li>2. Hazmat information in results is as reported for the PortOfCall/Non-EU departure while Hazmat information in results is as reported for the NextPort/EU-departure. The same holds for Crew&amp;Passengers.</li> <li>3. Voyages with status "On-hold" or "Dummy" are not provided in the results</li> </ol>

Use Case Req ID	<b>UC-SSN-SCREQ-21</b>	
Use Case Name	Request ShipCall from the Web	
Purpose	The current Use Case describes the functionality related to the search for Voyages from the Web interface.	
Subsystem	SSN Web application	
Primary Actor(s)	SSN User	
Precondition(s)	<p>The Actor has been authenticated in the system. At least one vessel with voyage(s) exist in the system. The actor is granted the permissionShipCall requests, based on the following:</p> <p>On the SHIPCALL_REQUESTOR task and geographical restrictions based on the Port Of Call.</p> <p>On the HAZMAT_REQUESTOR task if HazmatDetails are requested.</p> <p>On the SECURITY_REQUESTOR task if SecurityDetails are requested.</p> <p>On the WASTE_REQUESTOR task if WasteDetails are requested</p> <p>On the EXEMPTION_REQUESTOR task if Exemption information exists</p>	

	On the CREWPAX_REQUESTOR task if Crew and Passengers Details are requested.
Postcondition(s)	Voyage(s).
Trigger(s)	The Actor wants to find information for a voyage(s) of a vessel.
Use Case Description	Primary Workflow
Step 1	The user navigates to the web page for searching voyages.
Step 2	<p>The user selects a predefined menu item that corresponds to a ship call request of the types indicated hereunder. The user shall enter the criteria applicable per case:</p> <p><i>Relevant Voyages:</i> Mandatory parameters: IMONumber or MMSINumber. Result: List of up to 13 most relevant Voyages in relation to the query Timestamp.</p> <ul style="list-style-type: none"> <li>– With ETA/ATAPortOfCall after and closest to the query timestamp or</li> <li>– ATDPortOfCall and ATAPortOfCall before and closest to the query timestamp or</li> <li>– ETDLastPort before and closest to the query timestamp, ATAPortOfCall not null and ATDPortOfCall after and closest to the query timestamp or</li> <li>– ETDLastPort before and closest to the query timestamp and PortOfCall unknown.</li> <li>– Only summary data are provided. No request for additional details from the data provider.</li> </ul> <p><i>Expected Call Of Selected Ship:</i> Mandatory parameters: IMONumber or MMSINumber; Optional parameters: StartDateTime, GetHazmat, GetWaste, GetSecurity, GetCrewPax. Business rules applicable [R4] Table 4– Description of queries supported by the MS2SSN_ShipCall_Req message. Result: Details of a Voyage</p> <ul style="list-style-type: none"> <li>– Without ATAPortOfCall, and</li> <li>– With ETAToPortOfCall after and closest to StartDateTime.</li> <li>– Hazmat/Security/Waste/Crew&amp;Pax information in results is as reported before departure from the port.</li> </ul> <p><i>Most Recent Arrival Of Selected Ship:</i> Mandatory parameters: IMONumber or MMSINumber; Optional parameters: StartDateTime, GetHazmat, GetWaste, GetSecurity, GetCrewPax. Business rules applicable [R4] Table 4– Description of queries supported by the MS2SSN_ShipCall_Req message. Result: Details of a Voyage</p>

- With ATAPortOfCall before and closest to StartDateTime, and
- With ATDPortOfCall, if available, after StartDateTime.
- Hazmat/Security/Waste/Crew&Pax information in results is as reported before departure from the port.

*MostRecentDepartureOfSelectedShip:*

Mandatory parameters: IMONumber or MMSINumber; Optional parameters: StartDateTime, GetHazmat, GetWaste, GetSecurity, GetCrewPax. Business rules applicable [R4] Table 4– Description of queries supported by the MS2SSN\_ShipCall\_Req message.

Result: Details of a Voyage

- Central SSN will provide the ship call with ATDPortOfCall before and closest to StartDateTime.
- Hazmat/Security/Waste/Crew&Pax information in results is as reported before departure from the port.

*RecentAndCurrentShipCallsOfSelectedShip:*

Mandatory parameters: IMONumber or MMSINumber; Optional parameters: StartDateTime, EndDateTime, NumberOfCalls. Business rules applicable [R4] Table 4– Description of queries supported by the MS2SSN\_ShipCall\_Req message.

Result: List of Voyages

- With ATAPortOfCall within the time period defined by StartDateTime and EndDateTime.
- If no time period is defined, Central SSN will provide the list of latest [NumberOfCalls] consolidated PortPlus messages with ATAPortOfCall before SentAt.
- Only summary data are provided. No request for additional details from the data provider.

*ExpectedShipCallsAtEUPort:*

Mandatory parameters: PortOfCall; Optional parameters: StartDateTime, EndDateTime, NumberOfCalls. Business rules applicable [R4] Table 4– Description of queries supported by the MS2SSN\_ShipCall\_Req message.

Result: List of Voyages

- With ETAToPortOfCall within the time period defined by StartDateTime and EndDateTime, and
- Without ATAPortOfCall.
- If no time period is defined, Central SSN will provide the list of [NumberOfCalls] correlated voyages:
- With ETAToPortOfCall after SentAt, and
- Without ATAPortOfCall.
- Only summary data are provided. No request for additional details from the data provider.



*CurrentShipCallsAtEUPort:*

Mandatory parameters: PortOfCall; Optional parameters: StartDateTime. Business rules applicable [R4] Table 4– Description of queries supported by the MS2SSN\_ShipCall\_Req message.

Result: List of Voyages

- With PortOfCall = defined PortOfCall, and
- With ATAPortOfCall after StartDateTime, and
- Without ATDPortOfCall.
- Only summary data are provided. No request for additional details from the data provider.

*CompletedShipCallsAtEUPort:*

Mandatory parameters: PortOfCall; Optional parameters: StartDateTime. Business rules applicable [R4] Table 4– Description of queries supported by the MS2SSN\_ShipCall\_Req message.

Result: List of Voyages

- With PortOfCall = defined PortOfCall, and
- With ATDPortOfCall after StartDateTime.
- Only summary data are provided. No request for additional details from the data provider.

*LatestCallUpdates:*

Mandatory parameters: StartDateTime, EndDateTime. Business rules applicable [R4]Table 4– Description of queries supported by the MS2SSN\_ShipCall\_Req message.

Result: List ofVoyages which were registered or updated within the specified time period.

- Only summary data are provided. No request for additional details from the data provider.

*ListExpectedCallsOfSelectedShip:*

Mandatory parameters: IMONumber or MMSINumber; Optional parameters: StartDateTime. Business rules applicable [R4] Table 4– Description of queries supported by the MS2SSN\_ShipCall\_Req message.

Result: List of Voyages

- Without ArrivalDetails element,and
- With ETAToPortOfCall after StartDateTime, and
- Without ATAPortOfCall.
- Only summary data are provided. No request for additional details from the data provider.

*SelectedShipCall:*

Mandatory parameters: ShipCallID; Optional parameters: GetHazmat, GetWaste, GetSecurity, GetCrewPax. Business rules applicable [R4] Table 4– Description of queries supported by the MS2SSN\_ShipCall\_Req

message.

Results:

a) a Voyage

- With the specified ShipCallID;
- Registered vessel IMO Number, MMSI Number, CallSign, ShipName, Flag;
- Hazmat/Security/Waste/Crew&Pax information in results is as reported before departure from the port.

b) list of notifications transmitted to SSN-EIS that refer to the same ship call.

*GetActiveHazmatForSelectedShip:*

Mandatory parameters: IMONumber or MMSINumber; Optional parameters: StartDateTime. Business rules applicable [R4] Table 4– Description of queries supported by the MS2SSN\_ShipCall\_Req message.

Result: the Voyage that is the most relevant and associated hazmat details which are active at StartDateTime.

- Information may come from different PortPlus messages (different values of ShipCallID).
- Hazmat information in results.

*GetActiveSecurityForSelectedShip:*

Mandatory parameters: IMONumber or MMSINumber; Optional parameters: StartDateTime. Business rules applicable [R4] Table 4– Description of queries supported by the MS2SSN\_ShipCall\_Req message.

Result: the most relevant Voyage and associated security details.

- Without ATDPortOfCall and With closest ETAToPortOfCall or ATAPortOfCall to StartDateTime
- Security information in results.

*GetActiveWasteForSelectedShip:*

Mandatory parameters: IMONumber or MMSINumber; Optional parameters: StartDateTime. Business rules applicable [R4] Table 4– Description of queries supported by the MS2SSN\_ShipCall\_Req message.

Result: the most relevant Voyage and associated waste details.

- Without ATDPortOfCall and
- With closest ETAToPortOfCall or ATAPortOfCall to StartDateTime
- Waste information in results. Details in case requested and are provided by the data provider. Summary in case requested or details are not provided by the data provider.

*GetActiveCrewPaxForSelectedShip:*

Mandatory parameters: IMONumber or MMSINumber; Optional parameters: StartDateTime. Business rules applicable [R4] Table 4– Description of queries supported by the MS2SSN\_ShipCall\_Req message.

	<p>Result: the most relevant Voyage and associated waste details.</p> <ul style="list-style-type: none"> <li>– Without ATDPortOfCall and</li> <li>– With closest ETAToPortOfCall or ATAPortOfCall to StartDateTime</li> <li>– Crew&amp;Pax information in results.</li> </ul>
Step 3	<p>The system searches the SSN database for a relevant voyage(s) based on the provided request criteria. The business rules for voyage retrieval defined in Annex A: Business Rules are also applicable.</p>
Step 4a	<p>In case of a request for a single voyage, the system displays a voyage which complies with the search criteria and the requestor's access rights and geographical restrictions.</p> <p>A magnifier glass – per case - indicates if the voyage has Hazmat and/or Security and/or Waste and/or Crew&amp;Passengers.</p> <p>The user clicks on a magnifier glass for the details.</p>
Step 4b	<p>In case of a request for a list of voyages the system displays the voyages that match the criteria and the requestor's access rights and geographical restrictions.</p> <p>Per record and per case a magnifier glass indicates if the voyage has Hazmat and/or Security and/or Waste and/or Crew&amp;Passengers.</p> <p>The user clicks on a magnifier glass for the details.</p>
Step 5	<p>The system identifies the relative information being Hazmat and/or Security and/or Waste and/or Crew&amp;Passengers.</p> <p>In case of the details where provided in V3 PortPlus the system identifies that the requested detailed information in XML must be acquired from an NCA Application (data provider). The system issues a ShipCall Request to the NCA application for the details.</p> <p>In case the details are Hazmat and where reported in v2 PortPlus notification the system will issue a v2 SSN2MS_ShipCall_Req. If URL the system will download the electronic file. In case of contact details it will extract from the database (skip step 6).</p>
Step 6	<p>The system receives within the timeout interval the response from the NCA application. It parses the QueryResults part of the response message.</p> <p>In case of a v2 MS2SSN_ShipCall_Res, the contents are mapped to the contents of the v3 SSN2MS_ShipCall_Res message.</p> <p>As a general rule regarding the SSN2MS_ShipCall_Res message:</p> <ul style="list-style-type: none"> <li>• The following elements are excluded (independent of the GetDetails request) from the message: "PurposeOfCall", "VesselDetails", "WasteConfirmation", "SecurityConfirmation", "CrewAndPaxConfirmation", "Exemptions".</li> <li>• Under element HazmatSummary: <ul style="list-style-type: none"> <li>○ If DGClassification = "BC" will be renamed to DGClassification = "IMSBC"</li> <li>○ If DGClassification = "INF", then the element "DG" will be ignored</li> </ul> </li> <li>• Element "Source" in "HazmatDetails" indicates the "From" and "SentAt" of the MS2SSN_ShipCall_Res.</li> <li>• Only one V3 "Consignement" element is created. It has no attributes "TransportDocumentID", "PortOfLoading",</li> </ul>

	<p>"PortOfDischarge".</p> <ul style="list-style-type: none"> <li>Each element "DPG" in "PlacementOfGoodsInContainer" of MS2SSN_ShipCall_Res is translated to element "DPGItem" with "TransportEquipmentUnit" element.</li> <li>Attribute "LocationOnBoardGoods" in element "PlacementOfGoodsInContainer" of MS2SSN_ShipCall_Res is copied to attribute "LocationOnBoard" in element "TransportEquipmentUnit" of each "DPGItem" element.</li> <li>Each element "DPG" in "PlacementOfGoods" of MS2SSN_ShipCall_Res is translated to element "DPGItem" with "NonTransportEquipmentUnit" element.</li> <li>Attribute "LocationOnBoardGoods" in element "PlacementOfGoods" of MS2SSN_ShipCall_Res is copied to V3 attribute "LocationOnBoard" in element "NonTransportEquipmentUnit" of each "DPGItem" element.</li> <li>As regards "DPGItem" elements: <ul style="list-style-type: none"> <li>A dummy value = "X" will be given to HazmatDetails&gt; CargoInformation&gt; Consignment&gt; DPGItem&gt; DGClassification</li> <li>Have no attributes "PackingGroup", "FlashPoint", "MarpolCode", "PackageType", "TotalNrOfPackages", "AdditionalInformation", "NoOfPackages".</li> <li>Attribute "TechnicalName" of MS2SSN_ShipCall_Res is translated "TextualReference".</li> <li>Have no element "EmS", "SubsidiaryRisks".</li> </ul> </li> </ul>
Step 7	<p>If the requestor has the access rights to requesting exemption data, the system searches for exemptions in the SSN database which are relevant based on the same rule as introduced in UC-SSN-SCREQ-20.</p> <p>If one or more exemption are found the details are fetched to be included in the details of the response to the user.</p>
Step 8	The system constructs the response to the user.
Step 9	The system displays the response.
Alternative Use Case Description	No relevant data found.
Step 1.1	The system identifies that no voyage satisfies the request criteria. The system prompts the user that no voyages were found.
Alternative Use Case Description	Exceeded Response Time/ Negative receipt
Step 2.1	If the response time (as defined in the request – see Special requirements) has been exceeded, or if a negative receipt has been received from the data provider's NCA Application, the system displays to the data requester with the summary that it holds in the database (notification details).
Input(s)	Request specific data.
Output(s)	Notification Details are displayed to the user.

Timer(s)	-
Business Process(es) Reference	-
Associated Use Case(s)	-
Special Requirements	Voyages with status "On-hold" or "Dummy" are not provided in the results

Use Case Req ID	<b>UC-SSN-SCRES-22</b>	
Use Case Name	<b>Process ShipCall Response</b>	
Purpose	Covers the functionality related to the systems' actions upon receiving a response for details from a Member State.	
Subsystem	SSN Core	
Primary Actor(s)	NCA Application, SSN GI	
Precondition(s)	The NCA Application (data provider) has received a request from SSN for providing detailed information which is declared available through a previously sent notification.	
Postcondition(s)	The system sends the response message with the requested information to the NCA (data requester) that initiated the request.	
Trigger(s)	Response message received from the NCA Application (data provider).	
Use Case Description	Primary Workflow	
Step 1	The system upon reception of a response message checks the associated response timer and the associated request message.	
Step 2	If the response period has not been exceeded, the system generates and sends the response to the Member State that initiated the request through XML message.	
Alternative Use Case Description	Response Time Exceeded	
Step 1.1	If the response time has been exceeded, the system shall only log the received response.	
Alternative Use Case Description	No matching Vessel Data	
Step 2.1	If the Vessels criteria defined in the response message do not match the vessel definition in the SSN data store then the system shall return a status code OK to the NCA application along with a warning that indicates the vessel particulars as defined in the SSN data store.	
Input(s)	Response message.	
Output(s)	Requested information.	
Timer(s)	Response Timer	
Business Process(es)	-	

Reference	
Associated Use Case(s)	Included UC: UC-SSN-MSG-01: Handle Incoming Message
Special Requirements	-

Use Case Req ID	<b>UC-SSN-EXF-23</b>	
Use Case Name	<b>Find Exemption</b>	
Purpose	The current Use Case describes the functionality related to the search for an exemption associated to a vessel in the Web interface.	
Subsystem	SSN Web Application	
Primary Actor(s)	SSN User	
Precondition(s)	The Actor has been authenticated in the system. At least one vessel with associated exemption has been inserted in the system. The actor is granted the permission EXEMPTIONS_REQUESTOR.	
Postcondition(s)	A list of exemptions.	
Trigger(s)	The Actor wants to find information for an exemption associated to a vessel.	
Use Case Description	Primary Workflow	
Step 1	The user navigates to the web page for searching exemptions.	
Step 2	The Actor searches for an exception by entering Vessel and type of Exemption as criteria.	
Step 3	The system displays a list of exemptions that match the criteria and the user is allowed to edit based on the geographical restrictions applied to the permission EXEMPTIONS_NOTIFIER applied to the field "Authority - Country" of the exemption.	
Step 4	The Actor selects an exemption from the list.	
Step 5	The system fetches from the data store and displays the Exemption details for the given ship.	
Alternative Use Case Description	Search an Exemption – No Results	
Step 1.5	No Exemption matches the search criteria. The system returns a related message and the flow continues from step 2 of the Primary Workflow.	
Input(s)	Search Criteria related to the vessel and exemption.	
Output(s)	Exemptions that satisfy the criteria are displayed.	
Timer(s)	-	
Business Process(es) Reference	-	

Associated Use Case(s)	-
Special Requirements	-

Use Case Req ID	<b>UC-SSN-MRSREQ-24</b>	
Use Case Name	<b>Process MRS Request (XML)</b>	
Purpose	Covers the functionality related to the system's actions upon receiving a request for Ship notification information through the System Interface	
Subsystem	SSN Core	
Primary Actor(s)	NCA Application	
Precondition(s)	A request for information has been received by the system.	
Postcondition(s)	The system has processed the request and the response message is provided to the NCA Application.	
Trigger(s)	A request for information has been sent to SSN in XML format	
Use Case Description	Primary Workflow	
Step 1	<p>The system performs the following authorization checks for the Actor submitting the request:</p> <ol style="list-style-type: none"> <li>1. The Actor is known to SSN application.</li> <li>2. The Actor has been activated.</li> <li>3. The Actor validity period has not elapsed yet.</li> <li>4. The Actor has been granted the required permissions.</li> <li>5. The Actor is authorized to submit requests from XML interface.</li> </ol>	
Step 2	<p>Depending on the Ship Notification Type (ShipNotType field) the system performs as follows:</p> <p>When <b>ShipNotType=AIS</b>,</p> <p>the system retrieves the latest AIS information available in SSN (on the basis of "Timestamp") for the specified vessel (identified by the "IMONumber", or the "MMSINumber" if "IMONumber" is not provided).</p> <p>If the attribute "SenderCountryId" (optional) is also provided, it will be used - in addition to previous criteria - to retrieve AIS information originating from that country; that is the sender's (domain attribute AisNotification-&gt;message-&gt;sender) country matching the given one.</p> <p>For the AIS Notification selected, the system will form a request (SSN2MS_Ship_Req) of type AIS, asking for additional information from the data provider (sender of AisNotification).</p> <p>When <b>ShipNotType=MRS</b>,</p> <p>the system retrieves the latest MRS information available in SSN (on the basis of "ReportingDateTime") for the specified vessel (identified by the</p>	

	<p>"IMONumber", or the "MMSINumber" if "IMONumber" is not provided).</p> <p>If the attribute "MRSIdentification" (optional) is provided, it will be used to track down the latest MRS Notification corresponding to this identifier only.</p> <p>If the attribute "SenderCountryId" (optional) is also provided, it will be used - in addition to previous criteria - to retrieve MRS Information originating from that country; that is the sender's (domain attribute MrsNotification-&gt;message-&gt;sender) country matching the given one.</p> <p>The matching MRS Notification can be of three kinds:</p> <ul style="list-style-type: none"> <li>• <b>V2 MRS Notification.</b> No detailed information can be obtained from the data provider (since V3 message exchange protocol is not supported), hence the system forms a reply to data requestor with the information locally stored, which in that case is only Notification data (no notification details).</li> <li>• <b>V3 MRS Notification received from Web interface.</b> Again no detailed information need to be obtained from the data provider; all information including MRS Notification Details (MrsNotificationDetails entity) are available in SSN, hence the system forms a full reply to data requestor.</li> <li>• <b>V3 MRS Notification received from XML interface.</b> The system needs to retrieve MRS Notification details from the data provider; thus, the system: <ul style="list-style-type: none"> <li>○ Creates an Additional Information Request (SSN2MS_Ship_Req) message and dispatches it to the corresponding Member State.</li> <li>○ Persists the request to an intermediate waiting queue. The messages will remain in the queue for a configurable time period or until a corresponding response is received from the data provider.</li> </ul> <p>The system will process the response returned from the Member State as described in "UC-SSN-MRSRES-26 - Process MRS Response".</p> </li> </ul>
Step 3	<p>Given that the system has processed the response returned from data provider, it has already created a reply (consolidated information including the initial request, the relevant notification and the detailed information from the data provider) stored in the waiting queue.</p> <p>It should be noted that SSN is not examining the response content. Even a negative response will be forwarded to data requestor.</p>
Step	<p>The system logs and sends the reply (SSN2MS_Ship_Res) to the data requestor.</p>
Alternative Use Case Description	<p>No matching Notifications found for the given request criteria</p>
Step 2.1	<p>The system creates a reply (SSN2MS_Ship_Res) which indicates that no notification were found.</p>



Step 2.2	The system logs the response message and sends it to the data requestor.
Alternative Use Case Description	No response has been received from the data provider (MS2SSN_Ship_Res) in the configured time period.
Step 3.1	The system retrieves from the intermediate waiting queue the request awaiting response from the data provider and forms a reply with the data available in the SSN central system; that is the MRS Notification data alone, no additional information from the data provider.
Step 3.2	The system logs and sends the reply (SSN2MS_Ship_res) to the data requestor.
Alternative Use Case Description	The Actor sending the request does not comply to authorization requirements
Step 1.1	The system creates a reply (SSN2MS_Ship_Res) which indicates that the Actor is not authorized to send the request.
Step 1.2	The system logs the response message and sends it to the data requestor.
Input(s)	Ship Request (MS2SSN_Ship_Req)
Output(s)	Ship Response (SSN2MS_Ship_Res)
Timer(s)	Response Timer
Business Process(es) Reference	-
Associated Use Case(s)	Included UC: UC-SSN-MSG-01: Handle Incoming Message Included UC: UC-SSN-MRSRES-26: Process MRS response
Special Requirements	

Use Case Req ID	<b>UC-SSN-MRSREQ-25</b>	
Use Case Name	<b>Request Ship Notification details from the Web</b>	
Purpose	The current Use Case describes the functionality related to the search for Ship notifications from the Web interface.	
Subsystem	SSN Web application	
Primary Actor(s)	SSN User	
Precondition(s)	The Actor has been authenticated in the system.The actor is granted the pertinent permission.	
Postcondition(s)	The request has been treated and the answer provided to the SSN user	
Trigger(s)	-	
Use Case Description	The Actor wants Ship Notification details.	
Step 1	The Actor navigates to the web page for requesting Ship Notification details (AIS/MRS Information -> Latest AIS/MRS for selected ship).	

Step 2	The system prompts the Actor to fill in the criteria to be used for requesting Ship Notification details (ShipNotificationDetailsRequest page).
Step 3	The system validates criteria according to the XML RG.
Step 4	<p>The system displays a list of Notifications matching the criteria.</p> <p>Depending on the Ship Notification Type (ShipNotType field) the results are collected as follows:</p> <p>When <b>ShipNotType=AIS</b>,</p> <p>the system retrieves the latest AIS information available in SSN (on the basis of "Timestamp") for the specified vessel (identified by the "IMONumber", or the "MMSINumber" if "IMONumber" is not provided).</p> <p>If the attribute "SenderCountryId" (optional) is also provided, it will be used - in addition to previous criteria - to retrieve AIS information originating from that country; that is the sender's (domain attribute AisNotification-&gt;message-&gt;sender) country matching the given one.</p> <p>When <b>ShipNotType=MRS</b>,</p> <p>the system retrieves the latest MRS information available in SSN (on the basis of "ReportingDateTime") for the specified vessel (identified by the "IMONumber", or the "MMSINumber" if "IMONumber" is not provided).</p> <p>If the attribute "MRSIdentification" (optional) is provided, it will be used to track down the latest MRS Notification corresponding to this identifier only.</p> <p>If the attribute "SenderCountryId" (optional) is also provided, it will be used - in addition to previous criteria - to retrieve MRS Information originating from that country; that is the sender's (domain attribute MrsNotification-&gt;message-&gt;sender) country matching the given one.</p>
Step 5	The Actor selects a Notification from the list.
Step 6	The system displays Notification information; includes only data maintained in SSN local store.
Step 7	The Actor requests details for the currently displayed Notification by clicking on the magnifier button under details.
Step 8	<p>Depending on the Ship Notification Type (ShipNotType field) the system performs as follows:</p> <p>When <b>ShipNotType=AIS</b>, the system:</p> <ul style="list-style-type: none"> <li>• forms a request (SSN2MS_Ship_Req) of type AIS, asking for additional information from the data provider (sender of AisNotification).</li> <li>• Persists the request to an intermediate waiting queue. The messages will remain in the queue for a configurable time period</li> </ul>

	<p>or until a corresponding response is received from the data provider</p> <p>The system will process the response returned from the Member State as described in "UC-SSN-MRSRES-26 – Process MRS Response". All data, including Notification details returned from data provider are temporarily stored in an intermediate storage (waiting queue).</p> <p>When <b>ShipNotType=MRS</b>, the currently displayed MRS Notification can be of three kinds:</p> <ul style="list-style-type: none"> <li>• <b>V2 MRS Notification.</b> No detailed information can be obtained from the data provider (since V2 message exchange protocol is not supported), hence the relevant option (magnifier button under details) is unavailable.</li> <li>• <b>V3 MRS Notification received from Web interface.</b> No detailed information need to be obtained from the data provider; all information including MRS Notification Details (MrsNotificationDetails entity) are available in SSN, hence the system stores the information (Notification details) temporarily stored in an intermediate storage (waiting queue).</li> <li>• <b>V3 MRS Notification received from XML interface.</b> The system needs to retrieve MRS Notification details from the data provider; thus, the system: <ul style="list-style-type: none"> <li>○ Creates an Additional Information Request (SSN2MS_Ship_Req) message and dispatches it to the corresponding Member State.</li> <li>○ Persists the request to an intermediate waiting queue. The messages will remain in the queue for a configurable time period or until a corresponding response is received from the data provider.</li> </ul> <p>The system will process the response returned from the Member State as described in "UC-SSN-MRSRES-26 – Process MRS Response".</p> <p>In every case, the requested data are placed in an intermediate queue, awaiting the web application to get them (poll for data).</p> <p>It should be noted that SSN is not examining the response content. A negative response, is a perfectly valid response and will be returned to the Actor's screen.</p> </li> </ul>
Step 9	The system redirects to a Ship Notification Details Response page, where it constantly (with a configurable interval) polls for the corresponding response.
Step 10	The system displays Notification details.
Alternative Use Case Description	Validation failed
Step 2.9	The system redirects to the Ship Notification Details Request page, where it clearly indicates the errors preventing the successful submission of the request.

Step.2.10	The flow continues from Step 2 of the main flow.
Alternative Use Case Description	Time-out occurred while awaiting for response.
Step 3.9	The system informs the Actor that the system wasn't able to collect the requested information in a timely fashion.
Input(s)	Request specific data.
Output(s)	Ship Notification Details are displayed to the Actor.
Timer(s)	-
Business Process(es) Reference	-
Associated Use Case(s)	Included UC-SSN-MRSRES-26: Process MRS response
Special Requirements	

Use Case Req ID	<b>UC-SSN-MRSRES-26</b>	
Use Case Name	<b>Process MRS Response</b>	
Purpose	Describes the system actions performed upon reception of a MRS response. The message is sent from the data provider as a response to previously sent request, for detailed MRS information, from SSN central application.	
Subsystem	SSN Core	
Primary Actor(s)	NCA Application	
Precondition(s)	A request for Ship Notification details has been accepted and SSN has sent a request for detailed MRS information to the data provider NCA Application (SSN2MS_Ship_Req)	
Postcondition(s)	The system has consolidated all the information for the MRS in an intermediate waiting queue ready to be dispatched to data requestor. This information includes, the request sent from requestor NCA application, the relevant notification in the SSN central application and the response returned from data provider's NCA application.	
Trigger(s)	The NCA Application has sent the response message (MS2SSN_Ship_Res)	
Use Case Description		
Step 1	The system validates the response against the XMLRG business rules.	
Step 2	The system associates the response with the corresponding request awaiting reply.	
Step 3	The system verifies that the response is compliant to the corresponding request (verifies the search criteria/vessel identification).	
Step 5	The system complements the corresponding MRS information stored in the waiting queue, with the detailed information provided in the	

	response.  In fact, the information dispatched to data requestor is the one returned from the data provider; locally stored information is not taken into account.
Alternative Use Case Description	No request found for the received response. This case may arise when a delayed response from the data provider is arrived, but the corresponding request has been removed due to timeout.
Step 2.1	The delayed response is silently discarded. No further SSN2MS_Ship_Res is sent to the initial; data requestor.
Alternative Use Case Description	Response is not compliant to corresponding request
Step 3.1	The response is silently discarded. No further SSN2MS_Ship_Res is sent to the initial; data requestor.
Input(s)	Response message.
Output(s)	Requested information.
Timer(s)	]
Business Process(es) Reference	-
Associated Use Case(s)	Included UC: UC-SSN-MSG-01: Handle Incoming Message
Special Requirements	-

Use Case Req ID	<b>UC-SSN-MRSLREQ-27</b>	
Use Case Name	<b>Process Ship List Request (XML)</b>	
Purpose	Covers the functionality related to the system's actions upon receiving a request for a list of MRS notification information for a specified geographical area in given time period through the system interface.	
Subsystem	SSN Core	
Primary Actor(s)	NCA Application	
Precondition(s)	A request for a list of information has been received by the system.	
Postcondition(s)	The system has processed the request and returned a response to data requestor.	
Trigger(s)	A request for a list of MRS notifications has been sent to SSN in XML format.	
Use Case Description	Primary Workflow	
Step 1	The system performs the following authorization checks for the Actor (Party) submitting the request: <ol style="list-style-type: none"> <li>1. The Party is known to SSN application.</li> </ol>	

	2. The Party has been activated. 3. The Party validity period has not elapsed yet. 4. The Party has been granted the required permissions. 5. The Party is authorized to submit requests from XML interface.
Step 2	The system validates the ship list request against the XMLRG business rules.
Step 3	The system retrieves all ShipNotifications corresponding to the criteria defined in the request; all MRS notifications where ReportingDateTime is between "StartDateTime" and "EndDateTime" If attribute "SenderCountryId" (optional) is used, the results are narrowed to notifications sent by a specific MS.
Step 5	The system forms a reply (SSN2MS_Ship_List_Res) for the given request. No request to the data providers is required. The system logs and sendsthe reply to the data requestor.
Alternative Use Case Description	No matching Notifications found for the given request criteria
Step 3.1	The system creates a reply which indicates that no MRS notification werefound.
Step 3.2	The system logs the response message and sends it to the data requestor.
Alternative Use Case Description	The Actor sending the request does not comply to authorization requirements
Step 2.1	The system creates a reply which indicates that the Party sending the request is not authorized to do so.
Step 2.2	The system logs the response message and sends it to the data requestor.
Input(s)	Ship List Request (MS2SSN_Ship_List_Req)
Output(s)	Ship List Response (MS2SSN_Ship_List_Res)
Timer(s)	-
Business Process(es) Reference	-
Associated Use Case(s)	Included UC: UC-SSN-MSG-01: Handle Incoming Message
Special Requirements	

Use Case Req ID	<b>UC-SSN-MRSLREQ-28</b>	
Use Case Name	<b>Process Ship List Request submitted from the Web</b>	
Purpose	Covers the functionality related to the system's actions upon receiving a	

	request for a list of MRS notification information for a specified geographical area in given time period submitted from the Web interface.
Subsystem	SSN Core / SSN Web Application
Primary Actor(s)	SSN User
Precondition(s)	The Actor has been authenticated in the system.The actor is granted the pertinent permission.
Postcondition(s)	The system has processed the request and returned a response to the Actor.
Trigger(s)	A request for a list of MRS notifications has been sent to SSN through information submitted from the SSN Web Application.
Use Case Description	Primary Workflow
Step 1	The user navigates to the web page for requesting a list of Ship Notifications (AIS/MRS Information > List of MRS notifications for a selected system).
Step 2	The system prompts the Actor to fillin the criteria to be used for requesting the List of Ship Notifications.
Step 3	The system validates the ship list request criteria against the XMLRG business rules.
Step 4	The system retrieves all ShipNotifications corresponding to the criteria defined in the request and renders the results in the same page, just below the given criteria.  All MRS notifications where ReportingDateTime is between "StartDateTime" and "EndDateTime"  If attribute "SenderCountryId" (optional) is used, the results are narrowed to notifications sent by a specific MS.
Step 5	The Actor selects any of the returned results and navigate to a page where Ship Notification information is displayed. It should be noted that only the notification information stored in the SSN central system is displayed; notification details originating from the data provider are not available.  The Actor may move back to results and select another Ship Notification to view available data.
Alternative Use Case Description	Validation fails for the criteria given by the user
Step 3.1	The Actor remains in the search criteria page, where the system clearly indicates errors preventing successful submission of the request. The Actor is prompted to correct them and re-submit the request.
Alternative Use Case Description	No matching Notifications found for the given request criteria.
Step 4.1	A corresponding message is displayed in Actor's screen and is prompted to attempt another search.
Alternative Use Case Description	Cancel Ship List Request.
5.1	Cancellation in any of the screens of the web flow, will clear search

	results and will let the Actor start the search over over.
5.2	The flow continues from step 2.
Input(s)	Ship List Request
Output(s)	Ship List Response
Timer(s)	-
Business Process(es) Reference	-
Associated Use Case(s)	
Special Requirements	

### 3.1.4 Monitoring Incident Report

This system package includes the services required to provide the end users with a textual interface to consult the IR notifications distribution status . This system package consists of the following use-case:

1. UC-SSN-IRM-31: Monitor IR notification status

Use Case Req ID	<b>UC-SSN-IRM-31</b>	
Use Case Name	<b>Monitor IR notification status</b>	
Purpose	Covers the functionality related to the consultation of the distribution status of IR notifications and IR feedback notifications (MS2SSN_IncidentDetail_Tx) in the Web interface.	
Subsystem	SSN Core, SSN Send Notifications console	
Primary Actor(s)	SSN User	
Precondition(s)	Actor is authorised for accessing to the system.	
Postcondition(s)	System provides the requested information.	
Trigger(s)	Actor requests the IR notification status.	
Use Case Description	Primary Scenario	
Step 1	The Actor navigates to SSN TI > Send notification > Incident Reports > Check distribution web console.	
Step 2	The Actor enters the search criteria: <ul style="list-style-type: none"> <li>– Incident ID (text search).</li> <li>– Incident type (single choice, default value: any). It applies to IR notifications only. If a incident type is selected, then only IR notifications are provided.</li> <li>– Recipient country (single choice, default value: any).</li> <li>– Date/Time From &amp; To of SentAt (default values: from [now-3 months] to [now]).</li> </ul>	
Step 3	The system returns a list of IR notifications distributed by the user's country. Information that will be displayed:	



	<ul style="list-style-type: none"> <li>– Type of message (IR notification / IR Feedback)</li> <li>– Incident Id</li> <li>– Incident Type</li> <li>– Sent At (default sort criterion, most recent on top)</li> <li>– Recipient countries (2-letter codes separated by commas)</li> <li>– Link to "Distribution status"</li> </ul>
Step 4	The Actor click on the "Distribution status" link and the application displays the "Distribution Acknowledgment status of Incident Report" and in addition general information regarding the Incident report (Incident ID, Incident Type, Sent At).
Step 4	The Actor select to export the results in a predefined file format: XLS, XML, CSV.
Alternative Use Case Description	No data found
Step 2.2	No IR notifications satisfy the search criteria.
Step 2.3	The system prompts the Actor with a message that no data were found.
Input(s)	Search criteria.
Output(s)	List of IR notification with distribution status.
Timer(s)	-
Business Process(es) Reference	-
Associated Use Case(s)	-
Special Requirements	-

### 3.1.5 MRS Management

This system package includes the services required to provide the end users with a textual interface to create, search and update MRS. This system package consists of the following use-cases:

1. UC-SSN-MRSMNG-41: Create MRS
2. UC-SSN-MRSMNG-42: Search/Update MRS

Use Case Req ID	<b>UC-SSN-MRSMNG-41</b>	
Use Case Name	<b>Create MRS</b>	
Purpose	Covers the functionality related to the create an MRS in the Web interface.	
Subsystem	SSN Application Management Console	
Primary Actor(s)	SSN User	
Precondition(s)	Actor is authorised for accessing to the system and for managing MRS	
Postcondition(s)	System provides the requested information.	

Trigger(s)	Actor creates a new MRS.
Use Case Description	Primary Scenario
Step 1	The Actor navigates to SSN Application Management > MRS Management > Create MRS console.
Step 2	The Actor enters the MRS identification and selects the countries assigned to the MRS. Once selected, the countries are listed below the MRS Identification field.
Step 3	The user clicks on the "Next" button.
Step 4	If there is a validation error, the user is prompted with the validation errors on the top of the page; he/she can alter the values and click again on "Next". If no errors the system displays the confirmation page.
Step 4	The Actor clicks on the "Submit" button to save the new MRS.
Input(s)	MRS identification and countries.
Output(s)	MRS defined in the database.
Timer(s)	-
Business Process(es) Reference	-
Associated Use Case(s)	-
Special Requirements	-

Use Case Req ID	<b>UC-SSN-MRSMNG-42</b>	
Use Case Name	<b>Search/Update MRS</b>	
Purpose	Covers the functionality related to the search/update an MRS in the Web interface.	
Subsystem	SSN Application Management Console	
Primary Actor(s)	SSN User	
Precondition(s)	Actor is authorised for accessing to the system and for managing MRS	
Postcondition(s)	System provides the requested information.	
Trigger(s)	Actor searches/updates an MRS.	
Use Case Description	Primary Scenario	
Step 1	The Actor navigates to SSN Application Management > MRS Management > Search/Update MRS console.	
Step 2	The Actor enters the search criteria: MRS Identification; Countries that assigned to the MRS. Click on "Search" button.	
Step 3	From the list of MRS that satisfy the criteria, click on the MRS	

	Identification to edit.
Step 4	The user may modify the MRS Identification (e.g. correct typos) or modify the set of countries tha may provide MRS information for the selected MRS Identification.
Step 5	The user clicks on the "Next" button.
Step 6	If there is a validation error, the user is prompted with the validation errors on the top of the page; he/she can alter the values and click again on "Next". If no errors the system displays the confirmation page.
Step 7	The Actor clicks on the "Submit" button to save the new MRS.
Alternative Use Case Description	Delete MRS
Step 2.5	The user clicks on the "Delete" button.
Step 2.6a	If the MRS is referenced in an MRS_NOTIFICATION, the system prompts the user that the MRS cannot be deleted.
Step 2.6b	If the MRS is not referenced an MRS_NOTIFICATION, the system prompts the user to confirm the deletion.
Input(s)	Search criteria; MRS Identification.
Output(s)	MRS definition is updated or deleted from the database.
Timer(s)	-
Business Process(es) Reference	-
Associated Use Case(s)	-
Special Requirements	-

## 4 Design of System Components

### 4.1 SSN-EIS

This section provides an overview of the SSN-EIS System. In this phase the current SSN system shall be re-factored to be SOA compliant and enhanced to expose more functionality (vessel, location and user) to external system via Web services.

A sovereign element in the design of the European Index Server (hereinafter EIS), as depicted in the logical view diagram presented in Figure 4-1, is the division of system in three distinguishable applications:

1. **ssn-core-app** (*ref:section 4.2*)
2. **ssn-console-app** (*ref:section 4.3*)
3. **ssn-xmlprotocol-app** (*ref:section 4.4*)

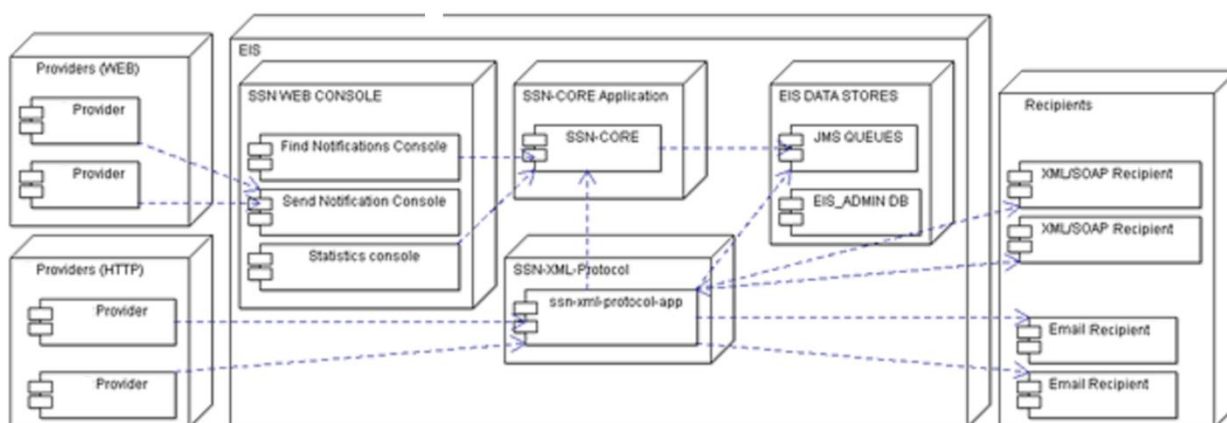


Figure 4-1 SSN EIS components connections diagram

The first application materialises the main functionality of EIS, which is the management of Messages, Vessels, Locations, Users in a way that is independent from the channel of communication through which messages are exchanged (HTTP/Html via the web console applications) or the natural representation of these messages (e.g. XML/SOAP via the SSN web XML application and web services respectively).

The second application provides the graphical user interface (GUI) and handling user-to-business requests. It is the presentation layer providing the functionality of the EIS system.

The third application, in its essence, is acting as a protocol adapter, providing the functionality of the main application, over the communication protocol specified in the XML Reference Guide; the supported protocols are

- HTTP; served by ssn-xmlprotocol-web application and
- SOAP; served by ssn-xmlprotocol-ws module exposes the web services.

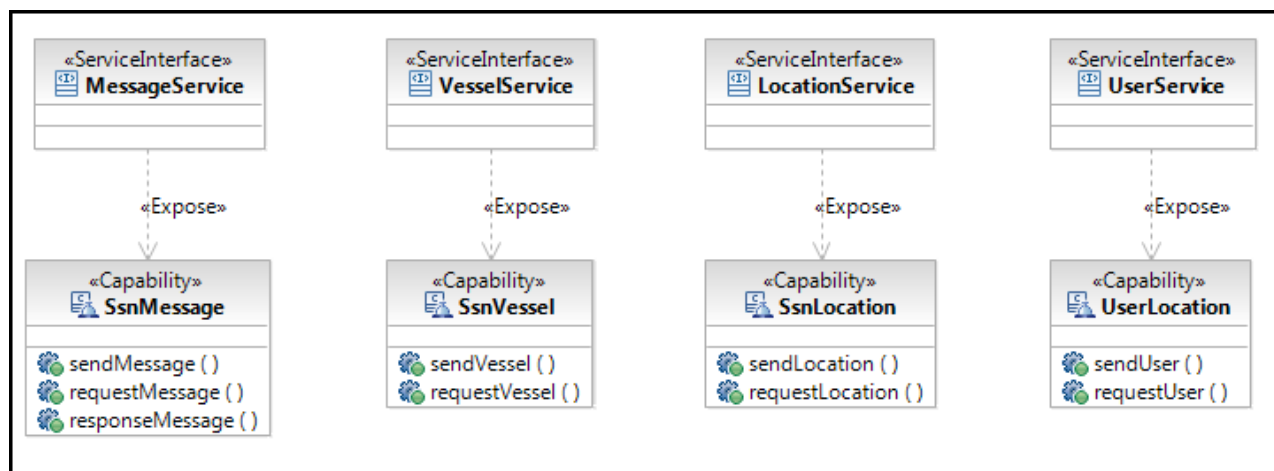
The decomposition of EIS in three applications allows for:

- The disengagement of the business logic of EIS from the protocol for which it is offered. This disengagement, allows also for the independent implementation of the business logic of EIS.
- The development of the communication protocol (e.g. changes in the XML Schema that specify the structure of messages) has local repercussions in the corresponding application and not in the entire system (EIS).

The SSN system is identified by the following primary entities:

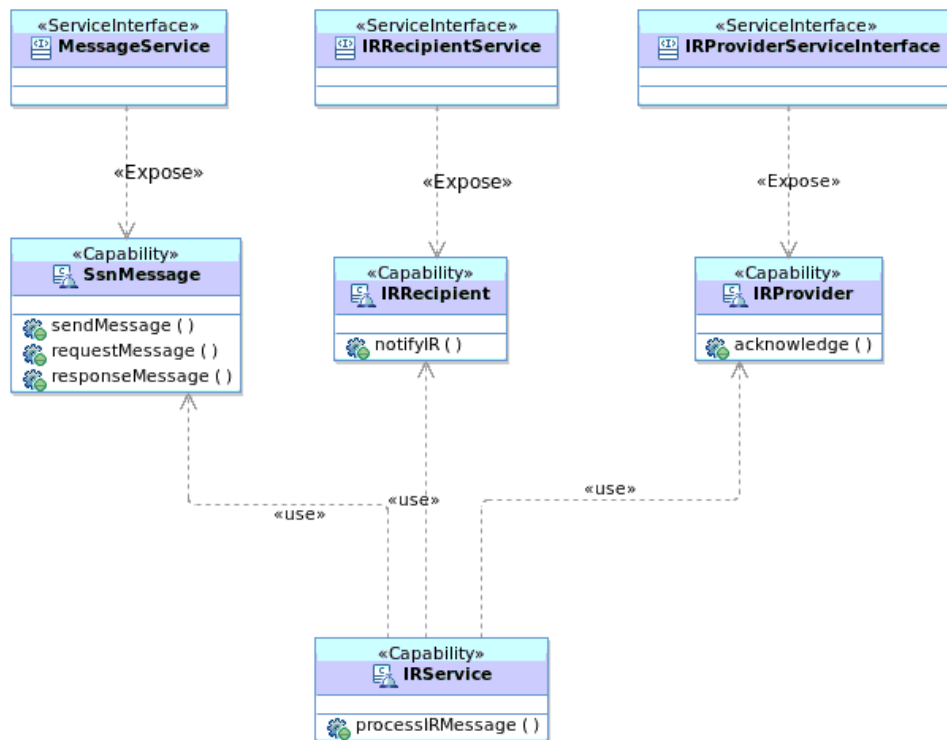
- "Message" as the primary artifact owned by the "Message Management" Business System.
- "Vessel" as the primary artifact owned by the "Vessel Management" Business System.
- "Location" as the primary artifact owned by the "Geography Management" Business System.
- "User" as the primary artifact owned by the "Organization Management" Business System.

Thus, SSN system provides a service that enables the access to, and update of, these entities as shown in the Figure 4-2.



**Figure 4-2SSN Servicesprovided operations**

In addition, the IR service is implemented that extends the existing MessageService as shown in Figure 4-6.



**Figure 4-3 SSN Services provided operations.**

Further to the core functionality of the system, independent business processes that carry out specific tasks are also deployed. Such a process is the LOCODE management from UNECE. This process is considered in section 4.13 of this document.

## 4.2 SSN Core Application - ssn-core-app

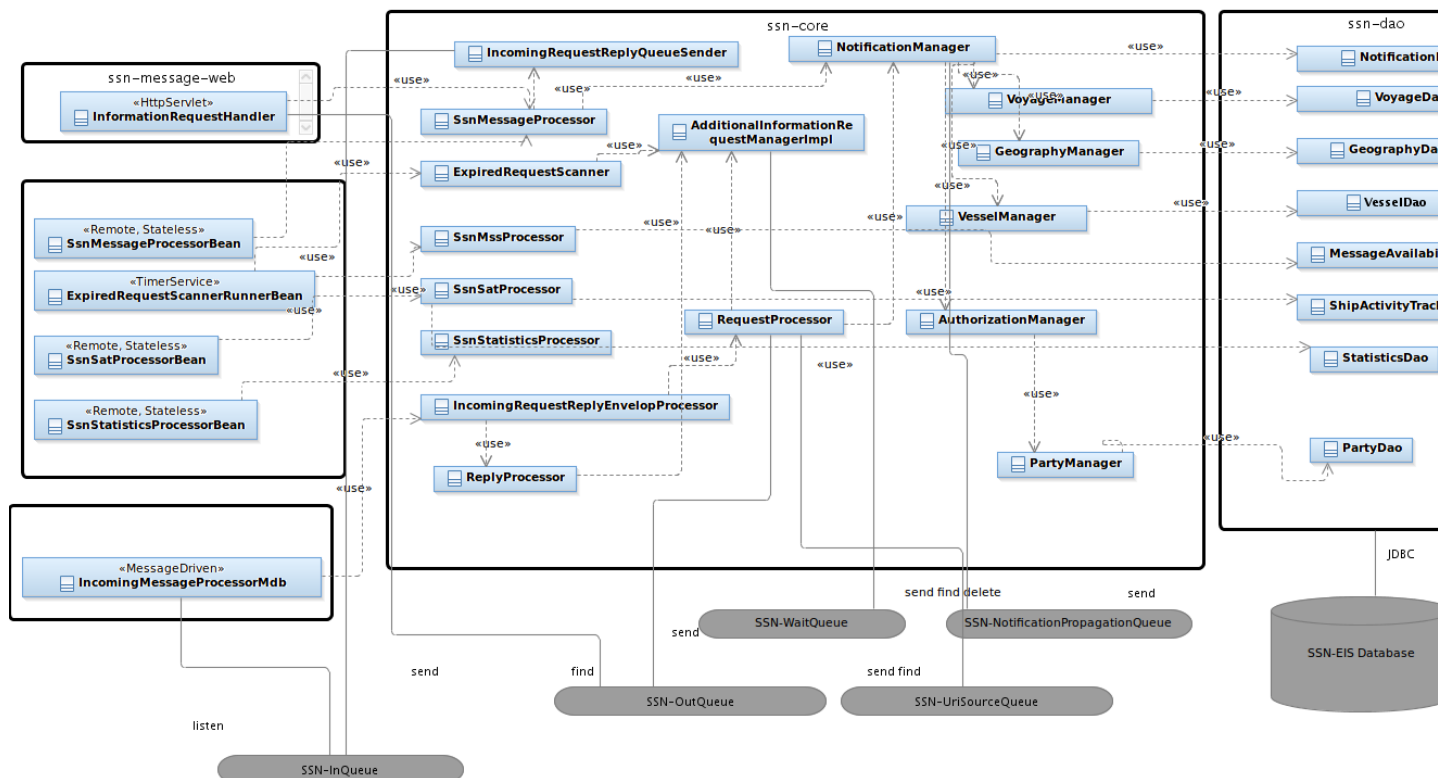


Figure 4-4 SSN Core Application - ssn-core-app

The application is constituted from the modules:

1. **ssn-domain**: SSN Business Domain Objects Modules. It includes the SSN Domain objects designed using Plain Java Classes and Interfaces. The SSN Domain objects encapsulate the state and behaviour of business entities. Examples of business entities in a SSN application are Vessels, Locations, Organizations, Notifications, etc. This module is used by all the other modules of SSN-EIS.
2. **ssn-support**: This module contains classes that provide support for e-mail creation, http sending, logging, management of SSN JMS queues and validation of SSN objects. This module is also used by all the other modules of SSN-EIS.
3. **ssn-dao**: This module implements the EIS data access. It *decouples application code from data access code*.
4. **ssn-core**: It is the heart of application. This module implements the SSN functionality.
5. **ssn-core-ejb**: It is a bundle of lightweight EJB's that lend Remote Access semantics to the main services of ssn-core, as well as asynchronous queue listeners (Message Driven Beans) and time services.
6. **ssn-message-web**: It implements a REST-based, HTTP interface to the application domain of EIS, by processing SSN message information requests and returning JSON formatted responses.

### 4.2.1 SSN Core Main Components

#### 4.2.1.1 SSN Domain module – ssn-domain

This module contains the domain model; the data object. This module is used by all the other modules of SafeSeaNet.

#### 4.2.1.2 SSN Support module - ssn-support

This module contains classes that are used by all the other modules of SafeSeaNet. It provides support for e-mail creation, http sending, logging, management of SSN JMS queues and validation of SSN objects. These classes are used by all the other modules of SSN-EIS

The basic components are listed in Table 4-1.

	Component	Description
1.	AbstractEnvelopProcessorMdb	An abstract processor implements the JMS Message Listener interface (onMessage() method).It provides the asynchronous processing of SSN Queues'messages.
2.	EmailManager	It provides the SSN Email functionality.
3.	HttpUtilityApacheImpl	It provides the SSN Http functionality; it is used to send and receive XML messages via HTTP to/from data providers/requestors.
4.	WebServiceClient	It provides the SSN "web service client" functionality; it is used to send SOAP messages via HTTP to data requestors.
5.	SsnPropertyPlaceholderConfigurer	It loads the application parameters (stored on EIS database) specified in the bean's definition.
6.	DelegatingValidator	An abstract implementation of the validator interfaces that resolves the SSN validator according to the object to be validated.

**Table 4-1 ssn-support**

#### 4.2.1.3 SSN Data Services module - ssn-dao

The basic components are listed in Table 4-3.

	Component	Description
1.	NotificationDao	Database data access object for the Notifications.
2.	AlertDistributionDao	Database data access object for the Distributed Alert Notifications.
3.	VoyageDao	Database data access object for the Voyages.
4.	ExemptionDao	A data access object for managing the database operations on Exemptions Information.
5.	PartyDao	Database data access object for the SSN users/authorities.
6.	VesselDao	Database data access object for the Vessels.
7.	GeographyDao	Database data access object for the Location codes.
8.	MessageAvailabilityDao	A data access object for managing the database operations for message Availability Configuration Items.
9.	StatisticsDao	A data access object for executing the statistical queries on EIS database.

**Table 4-2 ssn-dao**

#### 4.2.1.4 SSN Core module - ssn-core

The basic components are listed in Table 4-3.



	Component	Description
7.	SsnMessageProcessor	<p>It is a Facade for the synchronous management of incoming messages (see also Sequence Diagram: Process Message-Notification &amp; Request/Reply in section 4.2.2.3.4).</p> <p>The implementation simply delegates the actual message processing to one of the following processors:</p> <ul style="list-style-type: none"> <li>- NotificationManager for the synchronous processing of notifications</li> <li>- IncomingRequestReplyQueueSender for the processing of request/ reply messages.</li> </ul> <p>The SSN Message Processor synchronously replies with a ProcessResult on each incoming message processed.</p>
8.	SsnVesselProcessor	<p>It is a Facade for the synchronous management of incoming messages (see also Sequence Diagram: Process Vessel Notification in section 4.2.2.3.17).</p> <p>The implementation simply delegates the actual message processing to the VesselManager for the processing of vessel notifications and request messages.</p> <p>The SSN Vessel Processor synchronously replies with a ProcessResult on each incoming message processed.</p>
9.	SsnUserProcessor	<p>It is a Facade for the synchronous management of incoming messages (see also Sequence Diagram: Process User Notification/Request in section 4.2.2.3.14).</p> <p>The implementation simply delegates the actual message processing to the PartyManager for the processing of user notifications and request messages.</p> <p>The SSN User Processor synchronously replies with a ProcessResult on each incoming message processed.</p>
10.	SsnLocationProcessor	<p>It is a Facade for the synchronous management of incoming messages (see also Sequence Diagram: Process Location Notification in section 4.2.2.3.12).</p> <p>The implementation simply delegates the actual message processing to the GeographyManager for the processing of location notifications and request messages.</p> <p>The SSN Location Processor synchronously replies with a ProcessResult on each incoming message processed.</p>
11.	ProcessResult	<p>It models the result of the aforementioned SSN processing – the SSN_Receipt message.</p>
12.	NotificationManager	<p>It provides the synchronous management of Notifications received as well as the possibility to retrieve Notification that satisfies the criteria of an Information Request.</p> <p>The following rules apply to the way SafeSeaNet handle the Notifications received:</p> <ul style="list-style-type: none"> <li>- Notifications received will be recorded if technically correct. A technically correct notification must be a valid XML message, is compliant to the corresponding XSD,</li> </ul>

	Component	Description
		<p>the reported vessel, in case a vessel is reported, and satisfies the existing rules for the validation of the vessel IMO number, MMSI number, Ship Name and Call Sign and finally the reported location code is technically correct meaning that it consists of a 2 letter country code (according to ISO 3166) followed by a 3 character code element for the location name that will normally comprise three letters or the numerals 2-9.</p> <ul style="list-style-type: none"> <li>- The vessel data are used as input to the vessel repository. Inserts and updates of the vessel data in the vessel repository must adhere to the existing rules for managing the reference repositories and on the XML reference guide.</li> <li>- The location codes reported in the Notifications received can be used as input to the locations repository. A new location that is technically correct and has not previously been defined will be inserted in the locations repository and will be labelled. The system considers the technically correct locations reported in the Notifications received when performing an Area search or when filtering locations by Country in the Web application.</li> <li>- All the Notifications with technically correct vessel data that are recorded in the system can be queried by the SSN users when requesting for Notification details.</li> </ul> <p>It processes the incoming notifications using the following logic (see also Sequence Diagram : Notification Processing - Valid Notification in section 4.2.2.3.5):</p> <ul style="list-style-type: none"> <li>- The provided authorization manager is used to check the notification sender and his/her permissions.</li> <li>- The provided notification validator is used to validate the incoming notification.</li> <li>- The provided vessel manager is used to resolve the notification reported vessel data against to the vessel registry.</li> <li>- The provided geography manager is used to update the location registry with the technically correct locations reported in the Notifications received.</li> <li>- In case of PortPlus notification the provided notification dao used to update the notification registry if the notification refers to the same Ship Call (identified by a unique ShipCallId) (see also Sequence Diagram: Notification Processing – PortPlus Notification in section 4.2.2.3.5).</li> <li>- In case of Port, PortPlus, Ship and Hazmat notification the provided notification dao used to process the voyage calculation (see also Sequence Diagram: Notification Processing – Voyage Calculation in section 4.2.2.3.5).</li> <li>- The provided notification dao used to update the notification registry.</li> <li>- The provided alert distribution manager is used to manage the recipients and store the uploaded</li> </ul>

	Component	Description
		<p>document – if it is defined- of alert distribution</p> <ul style="list-style-type: none"> <li>- The provided notification propagation sender is used to notify the SSN users according to <ul style="list-style-type: none"> <li>o ship activity tracking services;</li> <li>o alert functionality upon receipt of an xml Alert notification.</li> </ul> </li> </ul> <p>It actually sends a message to the "notification propagation" JMS queue using the SsnMessageQueueSender interface.</p>
13.	IncomingRequestReplyQueue Sender	<p>It provides the management of the incoming xml Request or Additional Information Reply.</p> <p>Actually, it sends the incoming xml request/reply messages to the Incoming Queue via/using the SSN message Queue Sender.</p>
14.	SsnMessageQueueSender	<p>An interface responsible to provide access to the JMS queues.</p>
15.	IncomingRequestReplyEnvelopeProcessor	<p>It provides the asynchronous processing of incoming xml requests/replies. It delegates the request/reply message from the Incoming Queue to RequestProcessor/ReplyProcessor accordingly.</p>
16.	RequestProcessor	<p>It provides the management of incoming Information Request.</p> <p>It processes the incoming request using the following logic (see also Sequence Diagram: Process Request Reply from EIS in section):</p> <ul style="list-style-type: none"> <li>- The provided authorization manager is used to check the request sender and his/her permissions.</li> <li>- The provided notification manager is used to find whether there is a notification that satisfies the criterion of the given request. <ul style="list-style-type: none"> <li>o If there is not such notification, an information not found reply is created and sent to the requestor via the provided outgoingQueueSender.</li> <li>o If there is a notification, the corresponding reply builder is being asked to decide whether the SSN can form a reply on its own or whether SSN has to send a request for additional information to the member state that had sent the notification</li> <li>o If the SSN can form on its own the reply, then the reply builder is called to perform this task, and the reply is sent to the requestor via the outgoingQueueSender.</li> </ul> </li> </ul> <p>Otherwise, a relevant AdditionalInformationRequest is created and sent to the data provider via the</p>

	Component	Description
		<p>outcomingQueueSender and to the additionalInfomrationRequestManager.</p> <p>The following rules apply to the way SSN must handle the Requests for Notification details received with regards to the vessel criteria:</p> <ul style="list-style-type: none"> <li>- Requests for notification details with search criteria the vessel IMO and or MMSI numbers; if IMO and MMSI are provided the search is performed based on both the criteria for a matching pair: IMO and MMSI should match. The system will execute the query in the vessels' registry.</li> <li>- The Exemptions Information is taken into account – via the provided exemption manager- in case of Requests for Hazmat notification details.</li> </ul>
17.	outcomingQueueSender	It sends a message to the outgoing JMS queue using the SsnMessageQueueSender interface.
18.	AdditionalInfomrationRequest Manager	An interface responsible to provide access to the wait JMS.
19.	ReplyProcessor	<p>It provides the management of incoming replies from the data providers.</p> <p>It processes the incoming reply using the following logic (see also Sequence Diagram: Process Message- Reply in section4.2.2.3.4):</p> <ul style="list-style-type: none"> <li>- The additionalInfomrationRequestManager is used to find the waiting request made by SSN.</li> <li>- The appropriate reply builder – according to the initial data request – is used to create the reply.</li> <li>- The reply is sent to the requestor via the outcomingQueueSender</li> <li>- Finally, the additionalInfomrationRequest Manager is used to remove the processed the waiting request (additionalInformationRequest) from the wait queue.</li> </ul>
20.	VesselManager	<p>It provides the management of vessels (ships) and the logic of updating the vessel details based on the data from the Vessel and Message Notifications.</p> <p>The "resolve processing" of the incoming notifications' reported vessel data is the following (see also Sequence Diagram: Update Vessel Registry Using Reported vessel (IMO and MMSI Number, CallSign, Ship name and Flag in the case of PortPlus messages) in section 4.2.2.3.5):</p> <ol style="list-style-type: none"> <li>a. for vessel that can be resolved – based on the rules defined for the vessel identification/validation procedure (refer also to CS-0202 Vessel V&amp;V) – a reference (Foreign Key) will be made to the "Valid" or "Invalid" vessel;</li> <li>b. for a vessel that is not resolved– based on the rules defined for the vessel identification/validation procedure – a new "Temporary" vessel will be created and a</li> </ol>

	Component	Description
		<p>reference (Foreign Key) will be made to that vessel.</p> <p>The "Search Vessel" functionality, provided by the web console as the initial step of the "Send Notifications" interactive process, proposes ONLY the resolved – these are the vessels classified as "Valid". If the user wishes to send a notification for a new vessel, not registered in the EIS OVR yet, the user is able to enter the Vessel Identification attributes (IMO and MMSI plus CallSign, ShipName and Flag in the case of PortPlus messages). On notification submit, the aforementioned "resolve processing" is executed to create a new Temporary vessel record.</p> <p>Similarly the user is able – using the management console - to create a new vessel and /or update the vessel attributes (including the MMSI, CallSign, ShipName and Flag) identified by the IMONumber for a particular vessel version.</p>
21.	ExemptionManager	It provides the management of the Exemptions Information.
22.	ExpiredRequestScanner	<p>It offers the possibility of discovery and management of expired InformationRequest</p> <p>The "scan processing" is the following (see also Sequence Diagram: Process Expired Request from EIS in section4.2.2.3.9):</p> <ul style="list-style-type: none"> <li>- The additionalInfomrationRequestManager is used to search the wait queue for expired messages; the time living in the queue of an expired message is greater than its timeout value.</li> <li>- The expiredRequestProcessor is used to process the list of expired messages.</li> </ul>
23.	ExpiredRequestProcessor	<p>Processes an expired request that SSN has sent to a data provider.</p> <p>It processes the expired messages using the following logic:</p> <ul style="list-style-type: none"> <li>- An ExpiredRequestReply message is created to inform the requestor for the TimeOut occurred on the request processing.</li> </ul> <p>The reply is sent to the requestor via the outcomingQueueSender</p> <ul style="list-style-type: none"> <li>- Finally, the additionalInfomrationRequest Manager is used to remove the waiting request from the wait queue.</li> </ul>
24.	AuthorizationManager	It is a Facade for the authorisation of incoming xml messages. It uses the party manager
25.	PartyManager	It provides the user management, authentication, authorization.
26.	GeographyManager	It provides the management of locations and the logic of updating the location code repository with the location codes included in the Location and Message Notifications.
27.	AlertDistributionManager	It provides the management of Distributed Alert Notifications

	Component	Description
		and their details.
28.	StatisticsManager	It provides the management of EIS reporting.

**Table 4-3 ssn-core**

#### 4.2.1.5 SSN Core EJB module - ssn-core-ejb

The basic components are listed in Table 4-4.

	Component	Description
1.	SsnMessageProcessorEJB	Stateless Session Bean acting as proxy for the SsnMessageProcessor from ssn-core.
2.	SsnVesselProcessorEJB	Stateless Session Bean acting as proxy for the SsnVesselProcessor from ssn-core.
3.	SsnUserProcessorEJB	Stateless Session Bean acting as proxy for the SsnUserProcessor from ssn-core.
4.	SsnLocationProcessorEJB	Stateless Session Bean acting as proxy for the SsnLocationProcessor from ssn-core.
5.	SsnStatisticsEJB	Stateless Session Bean acting as proxy for the StatisticsManager from ssn-core.
6.	ExpiredRequestScannerRunnerBean	Time programmed Stateless Session Bean that periodically calls (on ejbTimeout method) the ExpiredRequestScanner in ssn-core. The interval duration of the timer is a system parameter; the SSN_INTERVAL_DURATIONApplication parameter defined in EIS database.
7.	IncomingRequestProcessorMDB	Message Driven Bean that calls the IncomingRequestReplyEnvelopProcessor in ssn-core in correspondence to the asynchronous reception of message from the Incoming Queue.

**Table 4-4 ssn-core-ejb**

## 4.2.2 UML Class and Sequence Diagrams

This section covers the architectural significant elements of the design model. It presents the definition of the most significant classes that will implement the requested functionality, organised into packages.

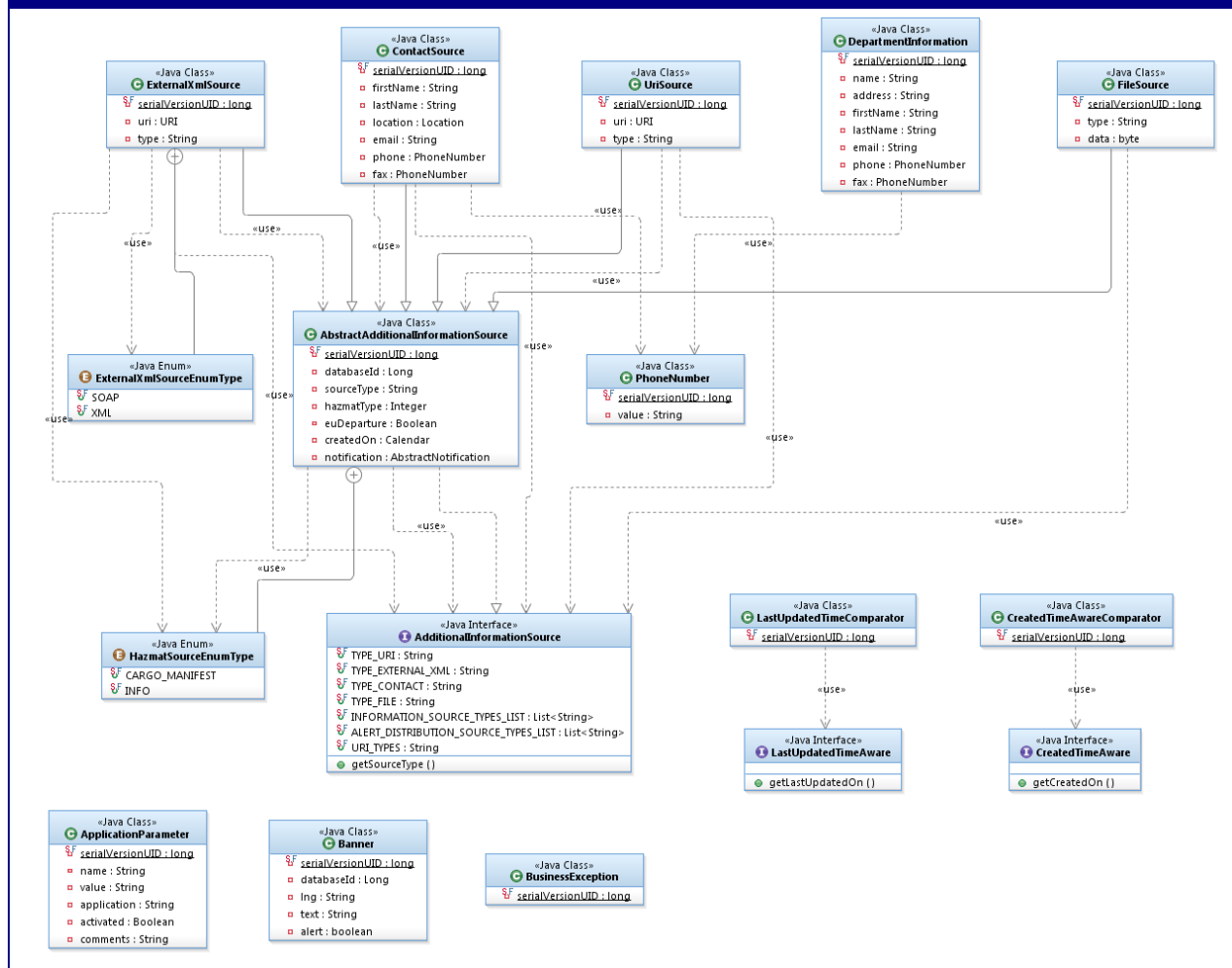
The UML Sequence Diagrams also presented in this section, associate classes and depict the overall flow of control within the system components.

The classes are organised in packages according to the functionality they provide. A package is a general-purpose model element that organizes model elements into groups. Each package contains a set of classes and interfaces, representing what will become components in the implementation.

#### 4.2.2.1 Module: ssn-domain

##### 4.2.2.1.1 Package: common

##### Class Diagram : common

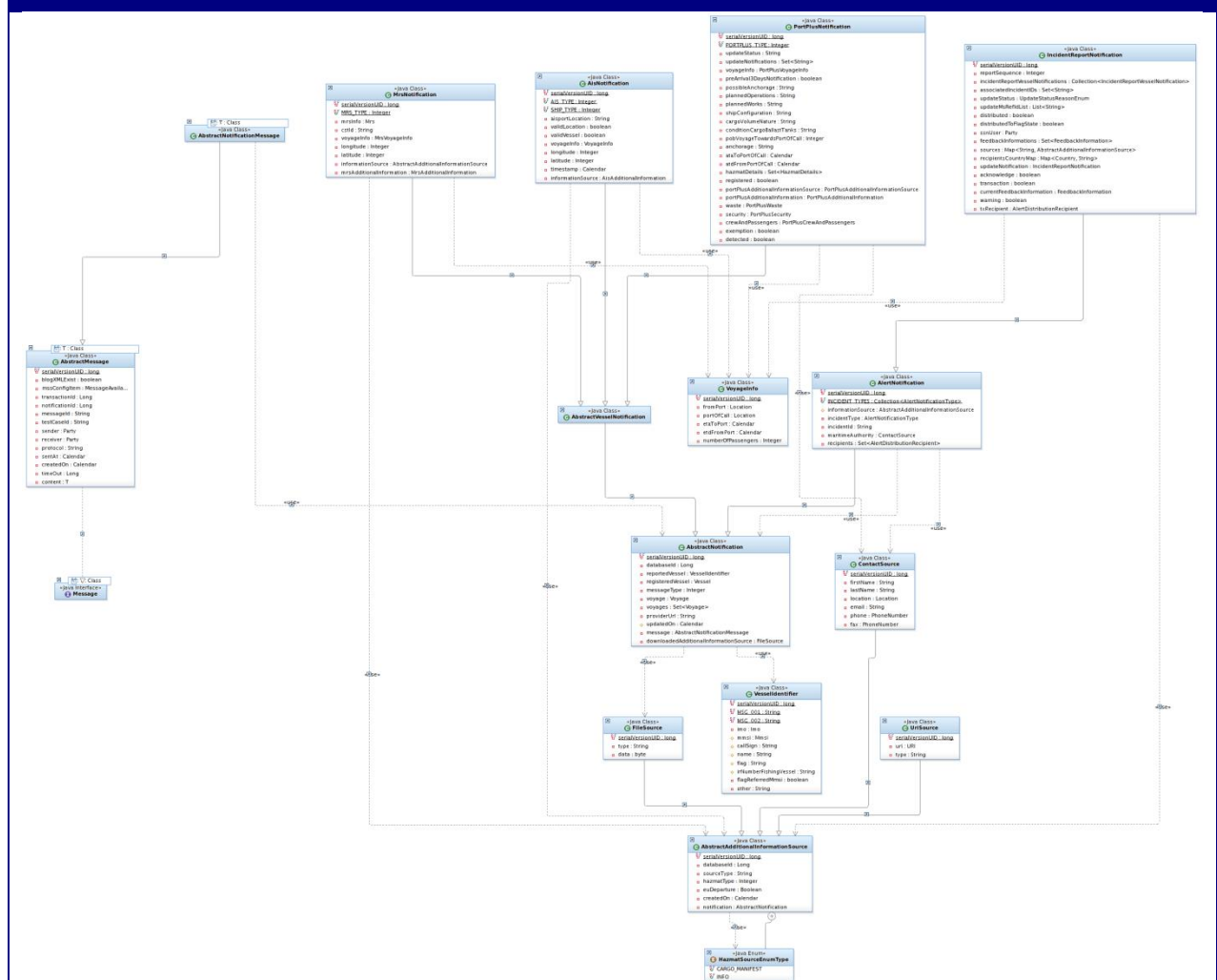


<b>Class</b>	<b>ApplicationParameter</b> Represents an application parameter such as the Environment (Training, Production, etc).
<b>Class</b>	<b>Banner</b> Represents the Ticket Banner text displayed on the Web console header.
<b>Class</b>	<b>BusinessException</b> The super class of all SSN business exceptions.
<b>Class</b>	<b>AbstractAdditionalInformationSource</b> Abstract implementation of the AdditionalInformationSource interface.
<b>Class</b>	<b>ContactSource</b> Represents a contact person's contact details (phone, fax etc).
<b>Class</b>	<b>PhoneNumber</b>

Class Diagram : common	
	This class represents a phone or fax number.
<b>Class</b>	<b>UriAdditionalInformation</b> Represents the encoded document holding the notification details.
<b>Class</b>	<b>UriSource</b> Represents the URL of a Web site where additional details can be accessed and the type of document the details can be downloaded in.
<b>Class</b>	<b>FileSource</b> Represents Additional information source file.
<b>Class</b>	<b>ExternalXmlSource</b> Represents Additional information that the data provided makes available upon request.
<b>Class</b>	<b>DepartmentInformation</b> This class represents the information about the department of an organization.
<b>Interface</b>	<b>AdditionalInformation</b> Defines an interface that has to be implemented by class that will process the contact details or the encoded document.
<b>Interface</b>	<b>AdditionalInformationSource</b> Defines an interface that has to be implemented by class that will decide the way details can be requested.
<b>Interface</b>	<b>CreatedTimeAware</b> An interface that should be implemented by persistent domain objects that want to know the time they have been created in the database.
<b>Interface</b>	<b>LastUpdatedTimeAware</b> An interface that should be implemented by persistent domain objects that want to know the last time they have been updated in the database.



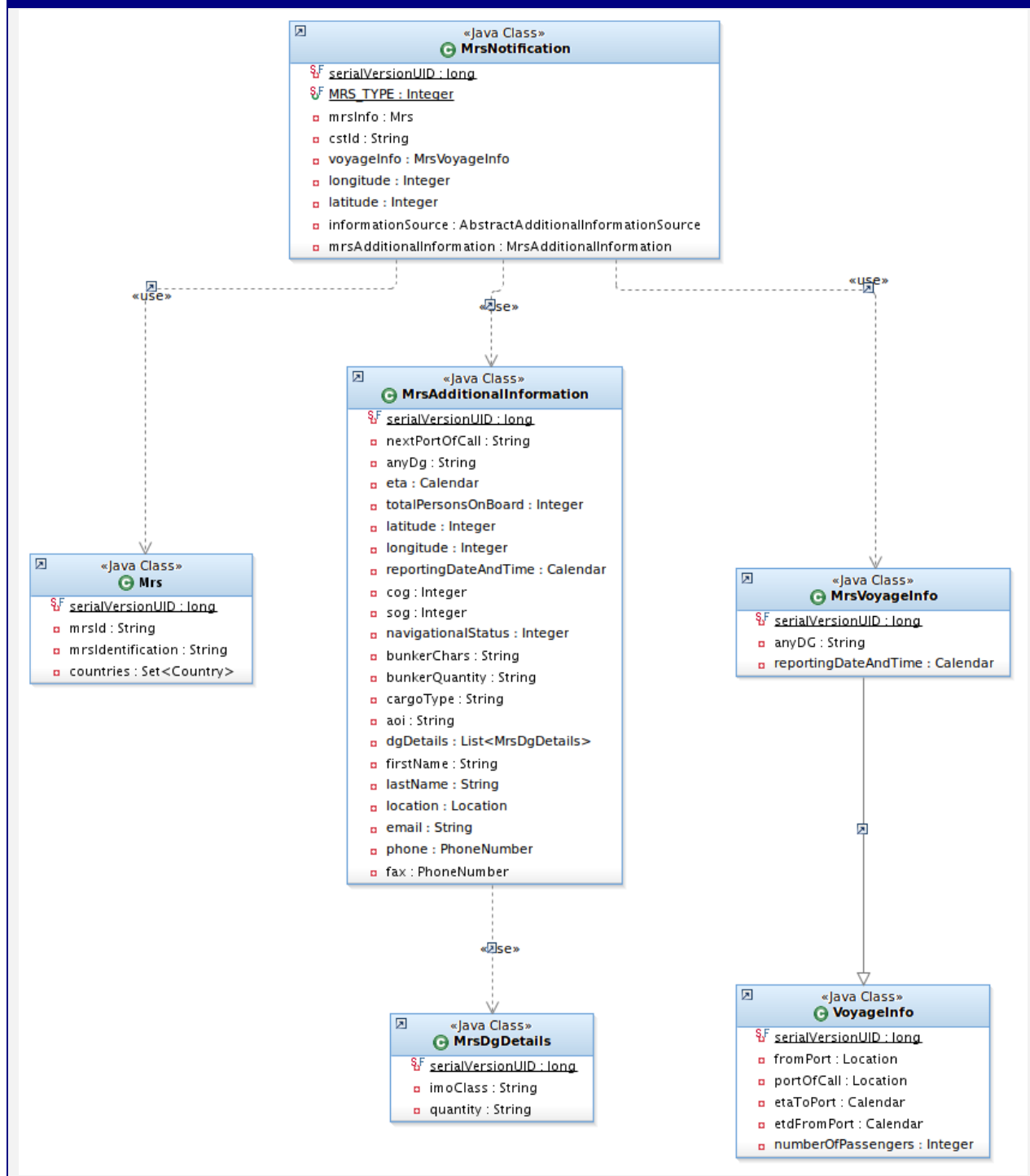
### Class Diagram : message & notification



<b>Interface</b>	<p><b><u>Message</u></b></p> <p>This interface represents all the messages that Safe Sea Net exchanges between the member states. It provides the methods/getters of common message attributes such as the message reference identifier, the sender, the time it was sent. The template parameter &lt;T&gt; represents the message content, such as notification, request, response.</p>
<b>Class</b>	<p><b><u>AbstractMessage</u></b></p> <p>An abstract implementation of the message interface</p>
<b>Class</b>	<p><b><u>AbstractNotification</u></b></p> <p>The notification information sent to SafeSeaNet by a Member State specifies the reported vessel, the registered (resolved on OVR) vessel.</p>
<b>Class</b>	<p><b><u>AbstractNotificationMessage</u></b></p> <p>This class extends the aforementioned AbstractMessage class to provide the methods/getters of common message attributes such as the reported and registered vessel. The template parameter &lt;T&gt;</p>

Class Diagram : message & notification	
	represents the notification information.
<b>Class</b>	<b>VesselIdentifier</b> This class represents the ship particulars reported in a notification. A provider can send a notification identifying a vessel either using an IMO number and/or a MMSI number. Additional attributes include: ShipName, CallSign and the vessel Flag.
<b>Class</b>	<b>FishingVesselIdentifier</b> This class extends the aforementioned VesselIdentifier class to represent the fishing vessel. Additional attributes include: IRNumber.
<b>Class</b>	<b>VoyageInfo</b> It holds the voyage information of a specific vessel such as the last port, the port of call, the number of passengers on-board.
<b>Class</b>	<b>AisNotification</b> AisNotification is a type of the ShipNotification. The notification is sent by a Member State to SafeSeaNet in order to notify SafeSeaNet about a vessel's voyage and cargo information. The ship notification in this case was originally captured via an AIS signal.

## Class Diagram : message & notification



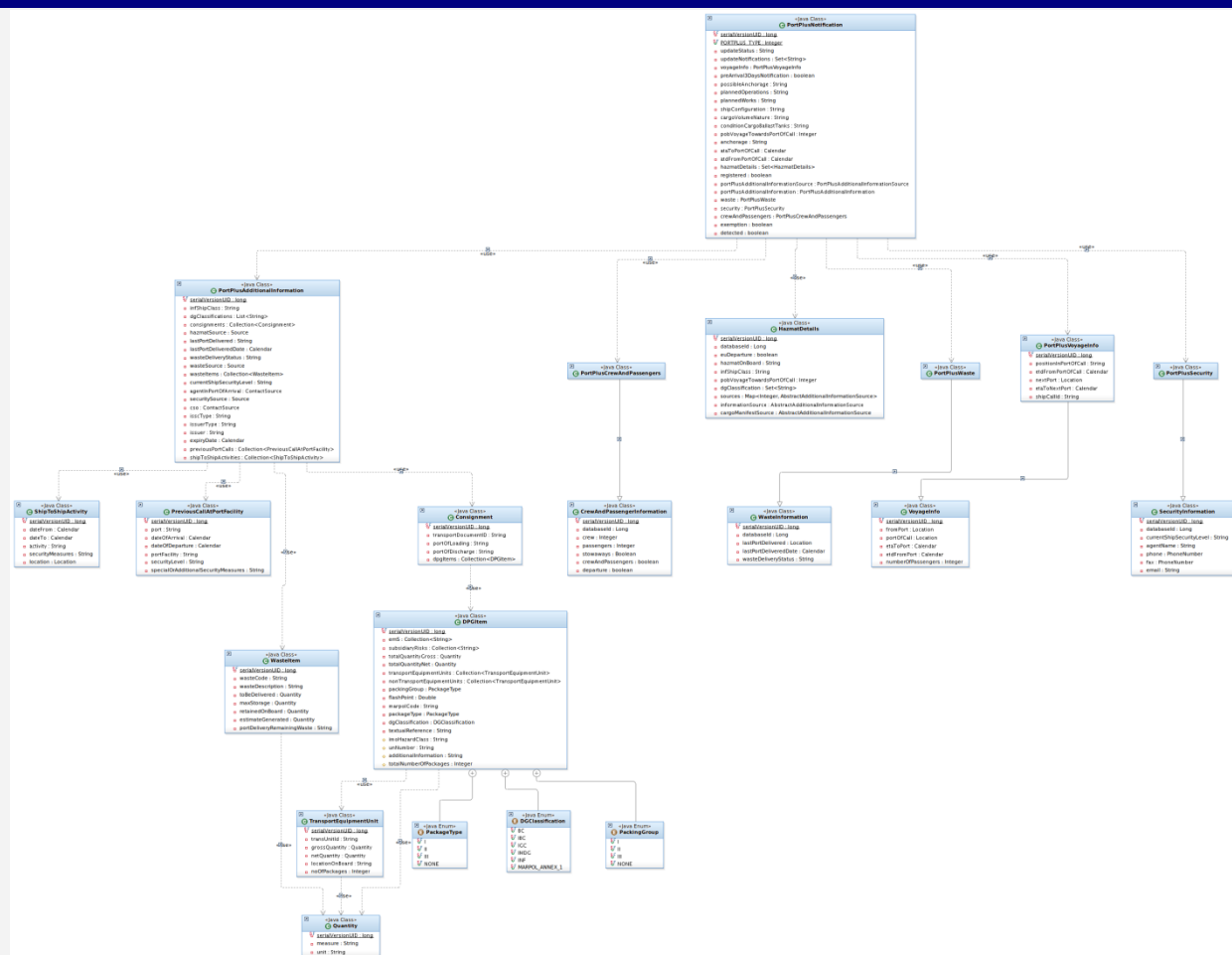
### Class

### MrsNotification

MrsNotification is a type of the ShipNotification.

The notification is sent by a Member State to SafeSeaNet in order to notify SafeSeaNet about a vessel's voyage and cargo information. The ship notification in this case was originally captured via an MRS signal.

## Class Diagram : message & notification



## Class

## PortPlusNotification

The notification is sent by a MemberState to SafeSeaNet in order to notify SafeSeaNet in case of

- 72h pre-arrival Notification
- 24h pre-arrival Notification
- Arrival Notification
- Departure Notification
- Hazmat information.
- Security information.
- Waste information.
- Crew and Passengers' information

The very 1<sup>st</sup> PortPlus notification received by SSN-EIS for a new ship call will create a new instance of a PortPlus notification in the SSN-EIS database. Every other PortPlus notification for the same ship call will not create a new instance of the PortPlus notification. Instead it will update the previously defined instance. This way the system consolidates automatically at notification process all the PortPlus notification data that refer to the same ship call. As such the system can directly service any request for details concerning a given ship call.

The PortPlus notification update process depends on the

## Class Diagram : message & notification

*NotificationStatus* element *UpdateStatus* attribute. Permitted values are:

- *UpdateStatus*="N"
- *UpdateStatus*="U"

The initial PortPlus notification for one ship call is expected with *UpdateStatus*="N".

Every consecutive PortPlus notification for the same ship call shall have *UpdateStatus*="U".

*UpdateStatus* rules include:

1. A ship call is uniquely identified by the *ShipCallId* attribute value. The *ShipCallId* is assigned by the data provider and in SSN-EIS must be unique per ship call per data provider. Shall the SSN-EIS system receive a PortPlus from the same data provider with an existing *ShipCallId* but for another vessel (identified by another *IMONumber* or another *MMSINumber* and *IMONumber* NULL) the message will be rejected with *StatusCode*="InvalidFormat" and *StatusMessage*="A *ShipCallId* has already been sent from user\_id\_xyz".
2. Shall the SSN-EIS system receive 2 or more PortPlus for the same ship call with *UpdateStatus*="N" then the 1<sup>st</sup> PortPlus will be accepted while all the others will be rejected.
3. Shall SSN-EIS receive a PortPlus with *UpdateStatus*="N" or *UpdateStatus*="U" with *PreArrival3Days* and/or *PreArrivalNotification24Hours* and/or *Arrival* and/or *Departure* notification details the message is valid and the update rules specified hereunder will be considered.
4. Shall the SSN-EIS system receive a PortPlus with *UpdateStatus*="U" then it should update the values of attributes – defined previously – by the values in the new PortPlus.

To cancel a value previously sent the data provider can send a PortPlus with the attribute value being NULL (e.g. *PossibleAnchorage*=""). In this case the previous value of *PossibleAnchorage* will be deleted in the database.

5. Shall the SSN-EIS system receive a PortPlus with *UpdateStatus*="U" for a new ship call (prior to the PortPlus with *UpdateStatus*="N") then the system must process the PortPlus. If at a later stage the system receives a PortPlus for the same ship call with *UpdateStatus*="N" the values of common attributes with previous PortPlus for the same ship will be those of the previous PortPlus and not of the latest one with *UpdateStatus*="N".

The system, upon receipt of the PortPlus with *UpdateStatus*="U" prior to the PortPlus with *UpdateStatus*="N" shall alert the data provider – instead of using an email - by the *SSN\_Receipt* with *StatusCode*="OK" and *StatusMessage*="The original PortPlus notification must be send".

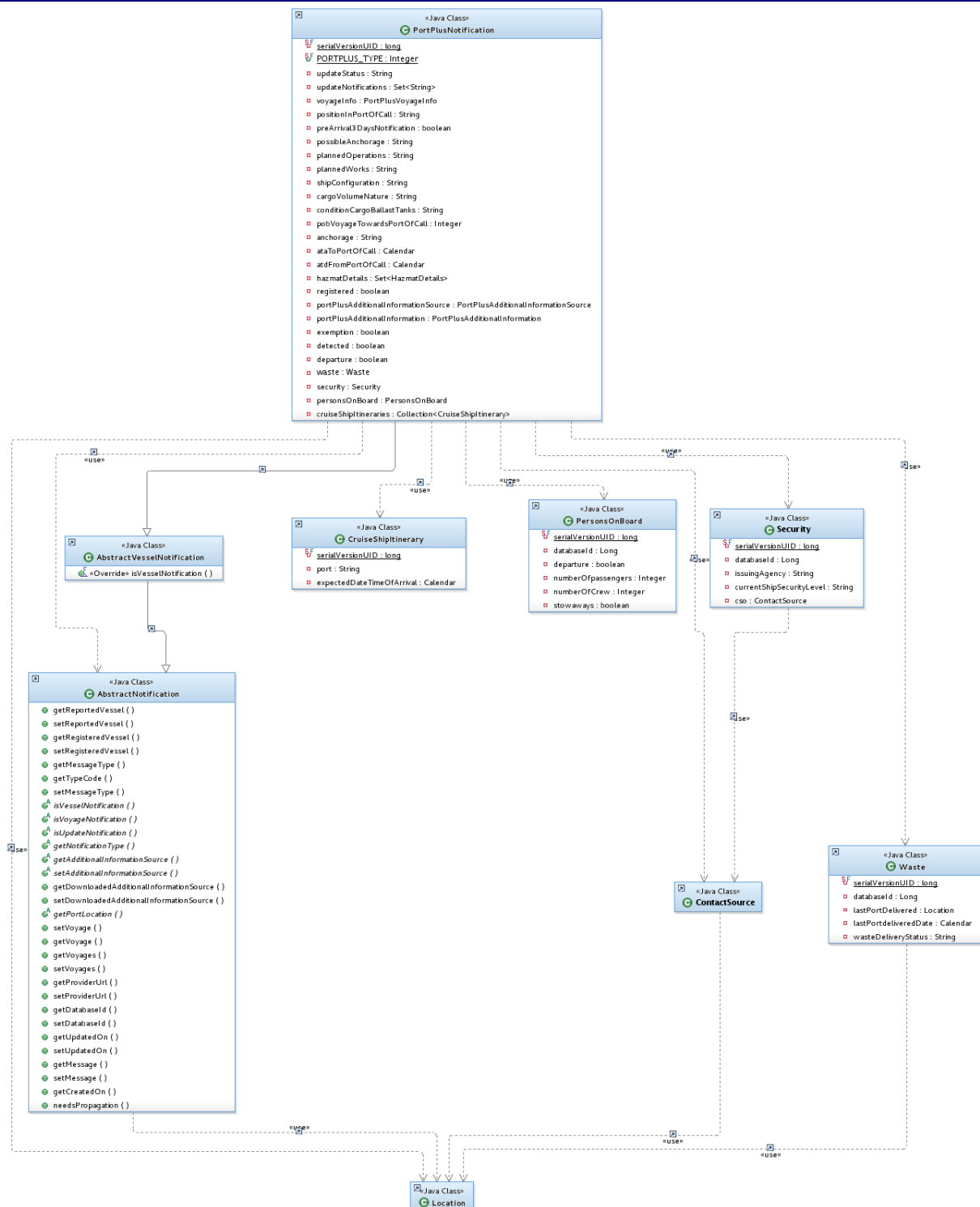
Please also refer to the following diagram for the specific PortPlus class diagram.

[illegible]

<b>Class</b>	<p><b><u>IncidentNotificationMessage</u></b></p> <p>This class extends the aforementioned AbstractNotificationMessage generic to represent the new Incident Report notification message.</p>
<b>Class</b>	<p><b>AlertNotification</b></p> <p>The notification information is sent by a Member State to SafeSeaNet in order to notify SafeSeaNet that the Member State holds some information about a specific incident type. It also holds the recipients.</p> <p>The notification can refer or not to a vessel.</p>
<b>Class</b>	<p><b>IncidentReportNotification</b></p> <p>It extends the Alert Notification to represent the information of new Incident Report notification such as the incident ID, the report sequence, the status, the incident details document – if any. Additional attributes include: the collection of the associated incident reports, the collection of the feedback information.</p>
<b>Class</b>	<p><b>IncidentReportVesselNotification</b></p> <p>It extends the aforementioned AbstractNotification class to represent the information of new Incident Report notification per</p>

Class Diagram : message & notification	
	associated vessel such as the vessel voyage information, the vessel positions at time of incident and reporting, the vessel cargo manifest.
<b>Class</b>	<b>IncidentPosition</b> It holds the position information of a specific vessel such as the position coordinates, the area, the bearing distance.
<b>Class</b>	<b>FeedbackInformation</b> It holds the feedback information of new Incident such as the authority reporting this action, the action details, the list of recipients.

## Class Diagram : PortPlus notification



### Class

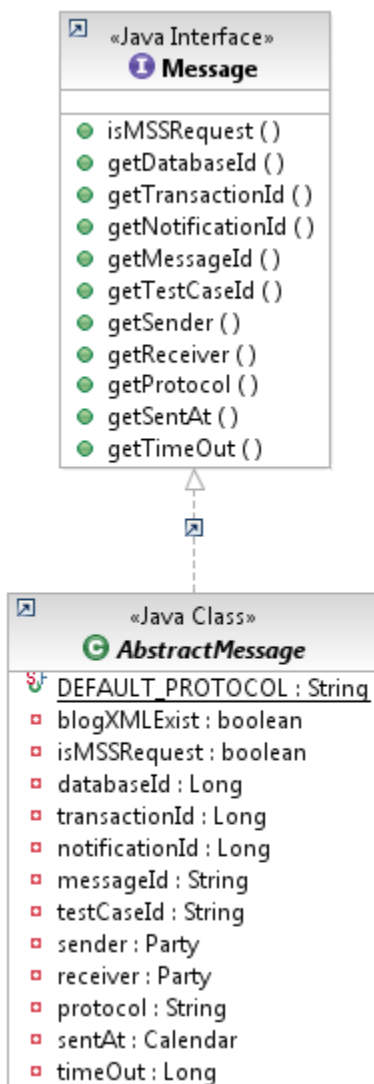
### Security

Security represents the generic CertificateType information provided by the NSW.



	<p>Attributes described bellow :</p> <p><b>issuingAgency</b> : Name of agency which issued the International Ship Security Certificate (ISSC)</p> <p><b>currentShipSecurityLevel</b> : Ship's current security level according to the ISPS standard</p> <p><b>cso</b>: The company security officer</p> <p>shipToShipActivities : The list of ship to ship activities</p> <p>lastTenPortCalls : The last ten port of calls.</p>
<b>Class</b>	<p><b>ShipToShipActivity</b></p> <p>This data type contains a list with the description of recent ship-to-ship activities and any security measures applied during these activities. The description shall contain the time and date when activity started and ended, the position and/or location (at least one), what activity was performed and if any special security measures were taken.</p> <p>Attributes:</p> <p>from/to/activity/additionalSecurityMeasures/location/ reportedLocationName/reportedPosition</p>
<b>Class</b>	<p><b>Waste</b></p> <p>This class implements contains information that shall be sent to a port in conjunction with an arrival.</p> <p>Attributes described bellow.</p> <p>lastPortDelivered : Last port where ship-generated waste was delivered. Name of the port where the generated waste of the ship was discharged.</p> <p>lastPortdeliveredDate : Last date when ship-generated waste was delivered Date indicating when the last disposal has been conducted.</p> <p>wasteDeliveryStatus : If ship delivers all, some or none of its waste in the port it reports to.</p>
<b>Class</b>	<p><b>PersonOnBoard</b></p> <p>Attributes described bellow:</p> <p>departure : Flag indicating whether the person on board information is for an arrival or for an departure notification.</p> <p>numberOfPassengers : Total number of passengers</p> <p>numberOfCrew : Total number of crew</p> <p>stowaways : Flag indicating whether the ship call reported any stowaways.</p>
<b>Class</b>	<p><b>WasteType</b></p> <p>Type of waste. The code shall be the one defined in Annex D. In addition, the proper shipping name is required for codes 504 (cargo residues) and all types of waste in category 2 (NLS).</p> <p>Otherwise, the text description of waste is optional.</p> <p>Attributes : Code/Description</p>

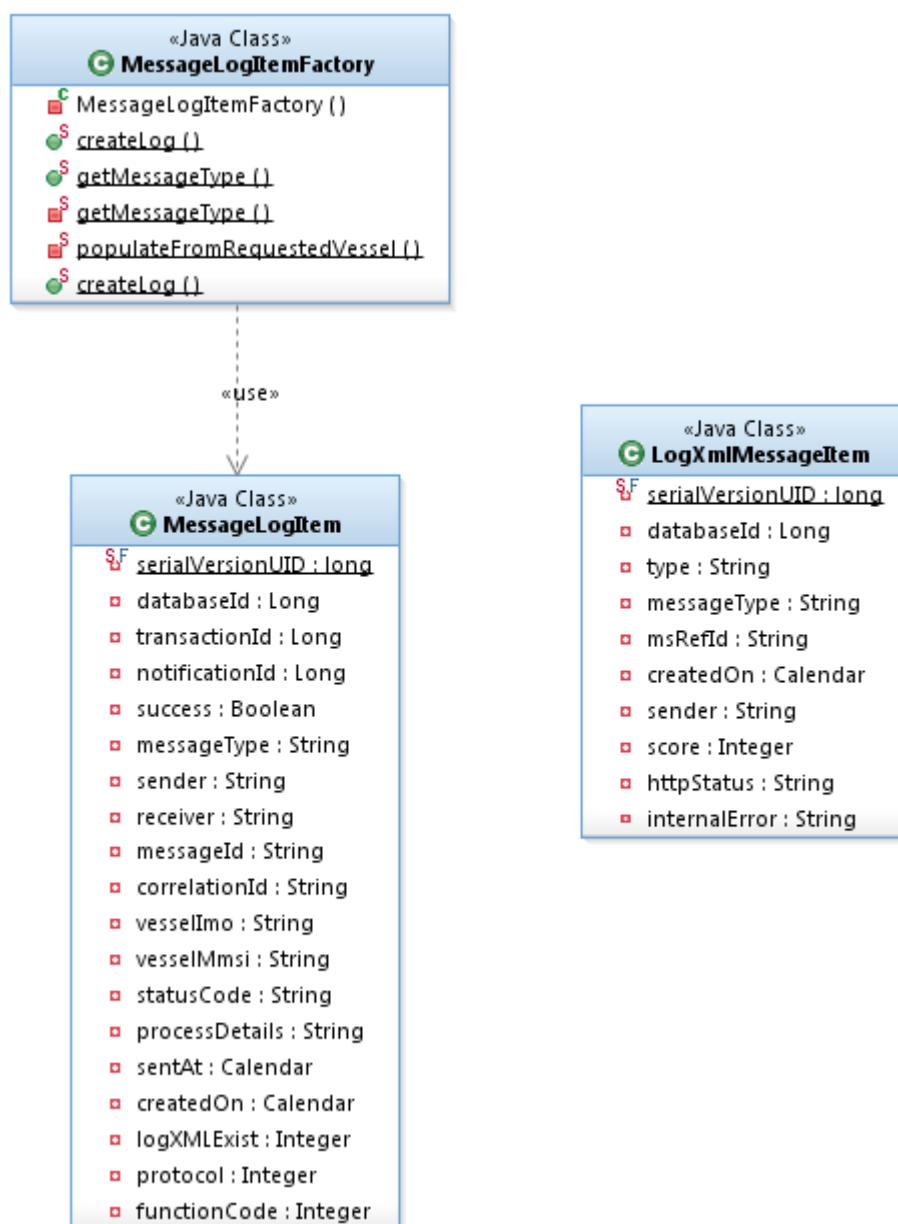
### Class Diagram : message



<b>Class</b>	<b><u>AbstractMessage</u></b> An abstract implementation of the message interface
<b>Interface</b>	<b><u>Message</u></b> Provides the methods/getters of message attributes.

#### 4.2.2.1.3 Package: message-log

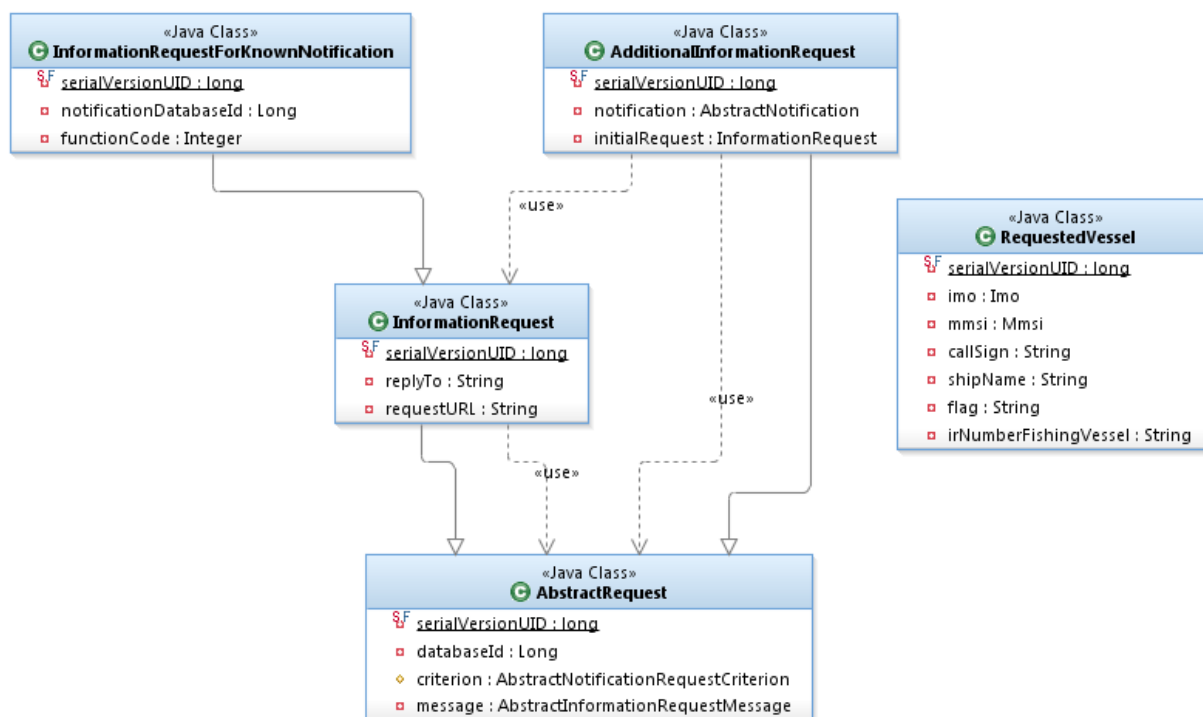
## Class Diagram : message-log



<b>Class</b>	<b><u>MessageLogItem</u></b> This class represents the entries logged on EIS database for the incoming messages (notification, request, reply) and the related process results. The attribute named <code>transactionId</code> identifies the Request/Response monitoring.
<b>Class</b>	<b><u>LogXmlMessageItem</u></b> This class represents the entries logged on EIS database for the incoming xml messages (MS notifications, requests, replies) and the related SSN xml responses (SSN replies/receipts).
<b>Class</b>	<b><u>MessageLogItemFactory</u></b> An abstract factory class used to create the message log items.

#### 4.2.2.1.4 Package: request

##### Class Diagram : request



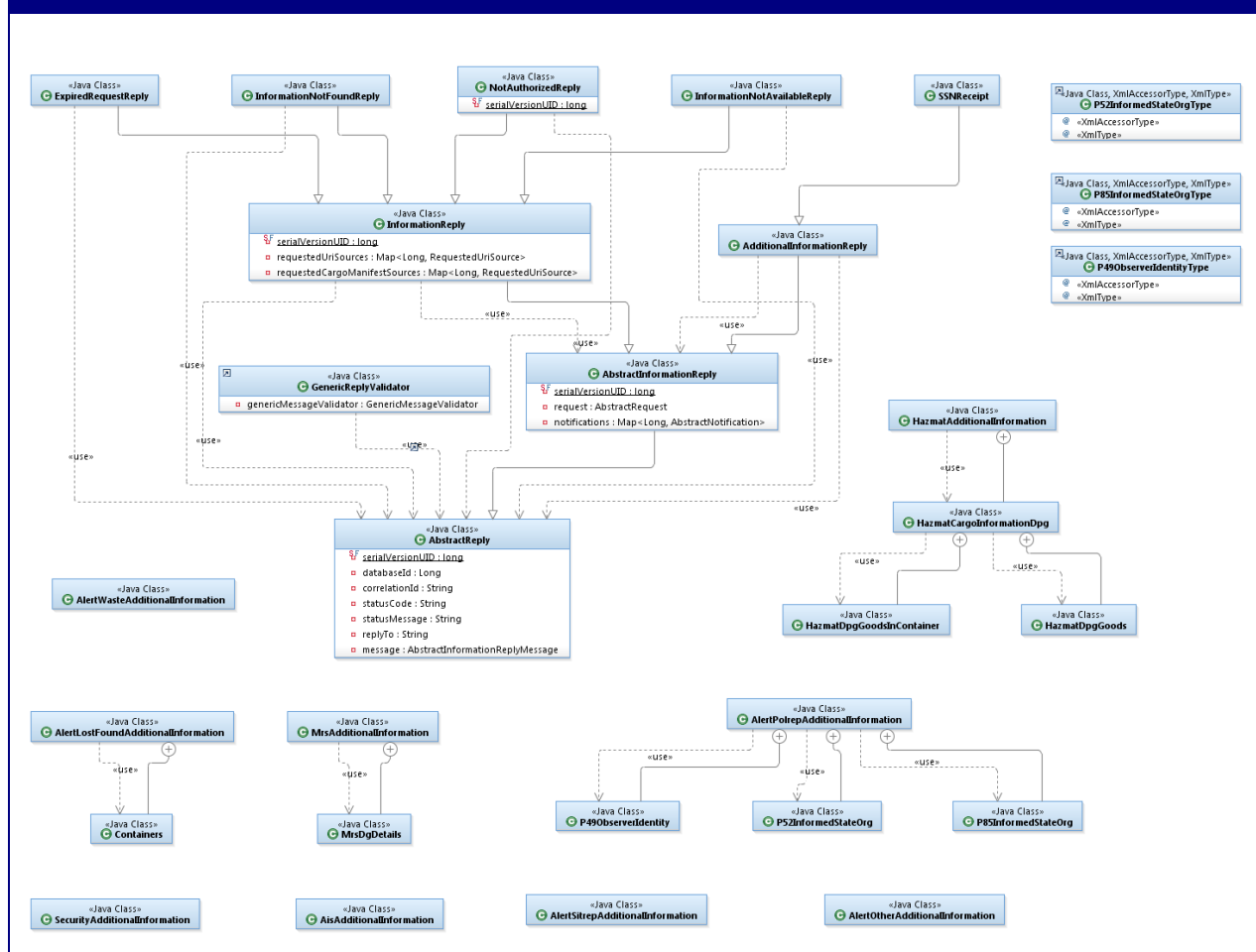
<b>Class</b>	<b>AbstractRequest</b> The request sent to SafeSeaNet by a MemberState specifies the sender, the time it was sent and the search criteria for identifying the vessel.
<b>Class</b>	<b>InformationRequest</b> The request sent by the member state specifies predefined search criteria for identifying the vessel (by IMO number or MMSI number when the vessel must be reported) and the type of notification for which to request details.
<b>Class</b>	<b>InformationRequestForKnownNotification</b> Extends InformationRequest for a request submitted by the ssn-core-console where the notification is already resolved.
<b>Class</b>	<b>AdditionalInformationRequest</b> The request for additional information details relating to a notification sent previously to SafeSeaNet is sent to the data provider member state that is expected to service the request with a response message.
<b>Class</b>	<b>RequestVessel</b> This class represents the vessel for which an Information Request is made. A requestor can send a request identifying a vessel either using an

**Class Diagram : request**

IMO number and/or an MMSI number.

#### 4.2.2.1.5 Package: reply

### Class Diagram : reply



<b>Class</b>	<b>AbstractReply</b> Abstract representation of a reply sent to a request
<b>Class</b>	<b>AbstractInformationReply</b> This class is the common parent of all classes that implement discrete reply messages.
<b>Class</b>	<b>InformationReply</b> Represents a reply sent by SSN to MS.
<b>Class</b>	<b>AdditionalInformationReply</b> Represents a reply from a data provider.
<b>Class</b>	<b>AisAdditionalInformation</b> The additional information of a ship (AIS) notification. This additional information is submitted by a MemberState to SafeSeaNet as a reply to a previous corresponding request from the SafeSeaNet.
<b>Class</b>	<b>AlertLostFoundAdditionalInformation</b> The additional information of an alert notification of type Lost/Found Containers. This additional information is submitted by a MemberState to SafeSeaNet as a reply to a previous corresponding

Class Diagram : reply	
	request from the SafeSeaNet.
<b>Class</b>	<b>AlertOtherAdditionalInformation</b> The additional information of an alert notification of type Other. This additional information is submitted by a MemberState to SafeSeaNet as a reply to a previous corresponding request from the SafeSeaNet.
<b>Class</b>	<b>AlertPolrepAdditionalInformation</b> The additional information of an alert notification of type POLREP. This additional information is submitted by a MemberState to SafeSeaNet as a reply to a previous corresponding request from the SafeSeaNet.
<b>Class</b>	<b>AlertSitrepAdditionalInformation</b> The additional information of an alert notification of type SITREP. This additional information is submitted by a MemberState to SafeSeaNet as a reply to a previous corresponding request from the SafeSeaNet.
<b>Class</b>	<b>AlertWasteAdditionalInformation</b> The additional information of an alert notification of type Waste. This additional information is submitted by a MemberState to SafeSeaNet as a reply to a previous corresponding request from the SafeSeaNet.
<b>Class</b>	<b>HazmatAdditionalInformation</b> The additional information of a Hazmat notification. This additional information is submitted by a Member State to SafeSeaNet as a reply to a previous corresponding request from the SafeSeaNet.
<b>Class</b>	<b>MrsAdditionalInformation</b> The additional information of a ship (MRS) notification. This additional information is submitted by a MemberState to SafeSeaNet as a reply to a previous corresponding request from the SafeSeaNet.
<b>Class</b>	<b>SecurityAdditionalInformation</b> The additional information of a Security notification. This additional information is submitted by a MemberState to SafeSeaNet as a reply to a previous corresponding request from the SafeSeaNet.
<b>Class</b>	<b>SSNReceipt</b> This information is submitted by a MemberState to SafeSeaNet as a reply to a previous corresponding request from the SafeSeaNet.
<b>Exceptional Replies</b>	
<b>Class</b>	<b>NotAuthorisedReply</b> Represents response messages send by SSN to the MS in reply to a request by a non-authorised party.
<b>Class</b>	<b>ExpiredRequestReply</b> Represents response messages send by SSN to the MS in reply to a timed out request.
<b>Class</b>	<b>InformationNotFoundReply</b> Represents response messages send by SSN to the MS in reply to a request for which no data was found.

#### Class Diagram : reply

##### Class

##### InformationNotAvailableReply

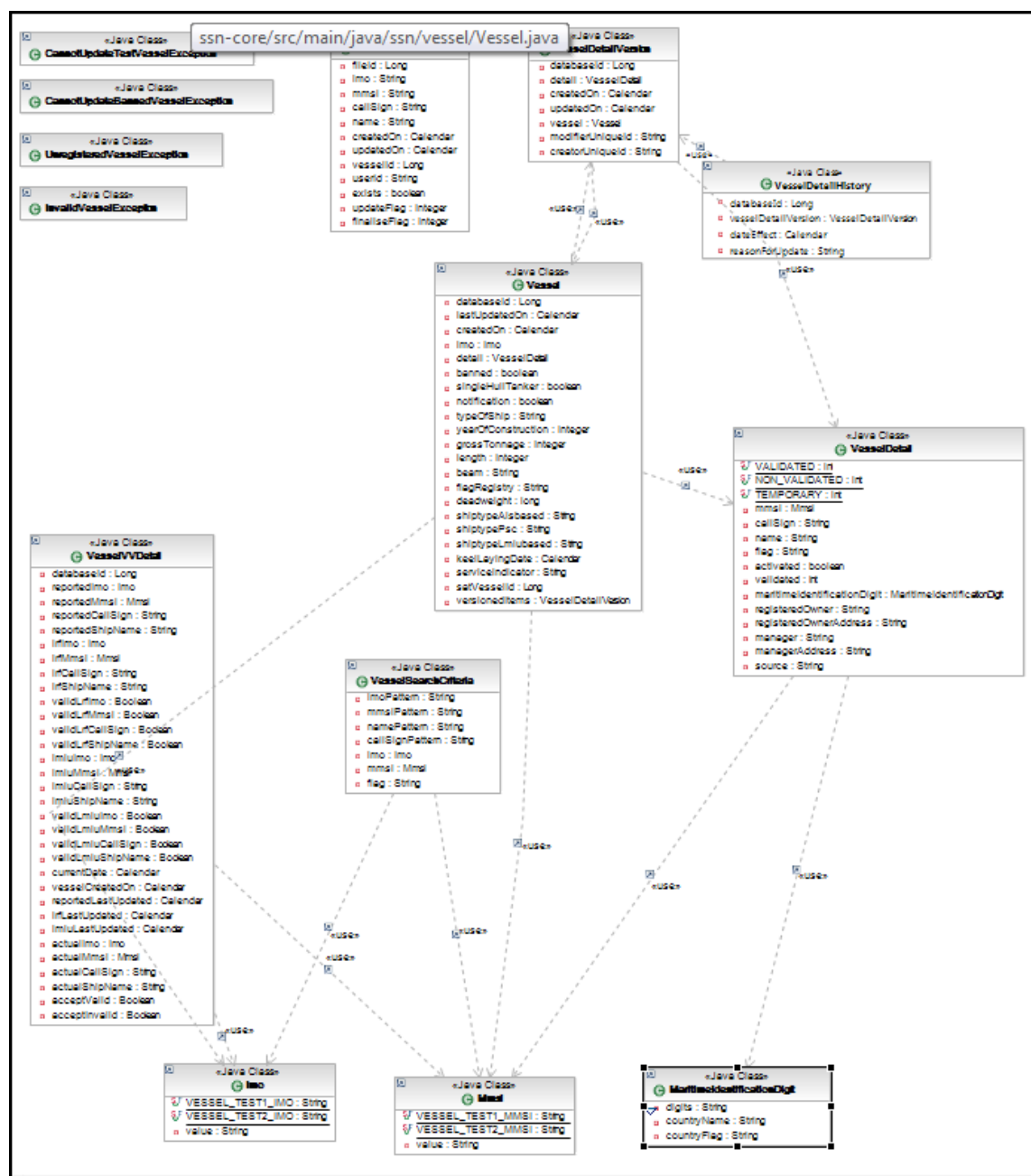
Represents response messages send by SSN to the MS in reply to a request for which no reply was received from the data provider.

#### 4.2.2.1.6 Package: vessel

#### Class Diagram : vessel



## Class Diagram : vessel



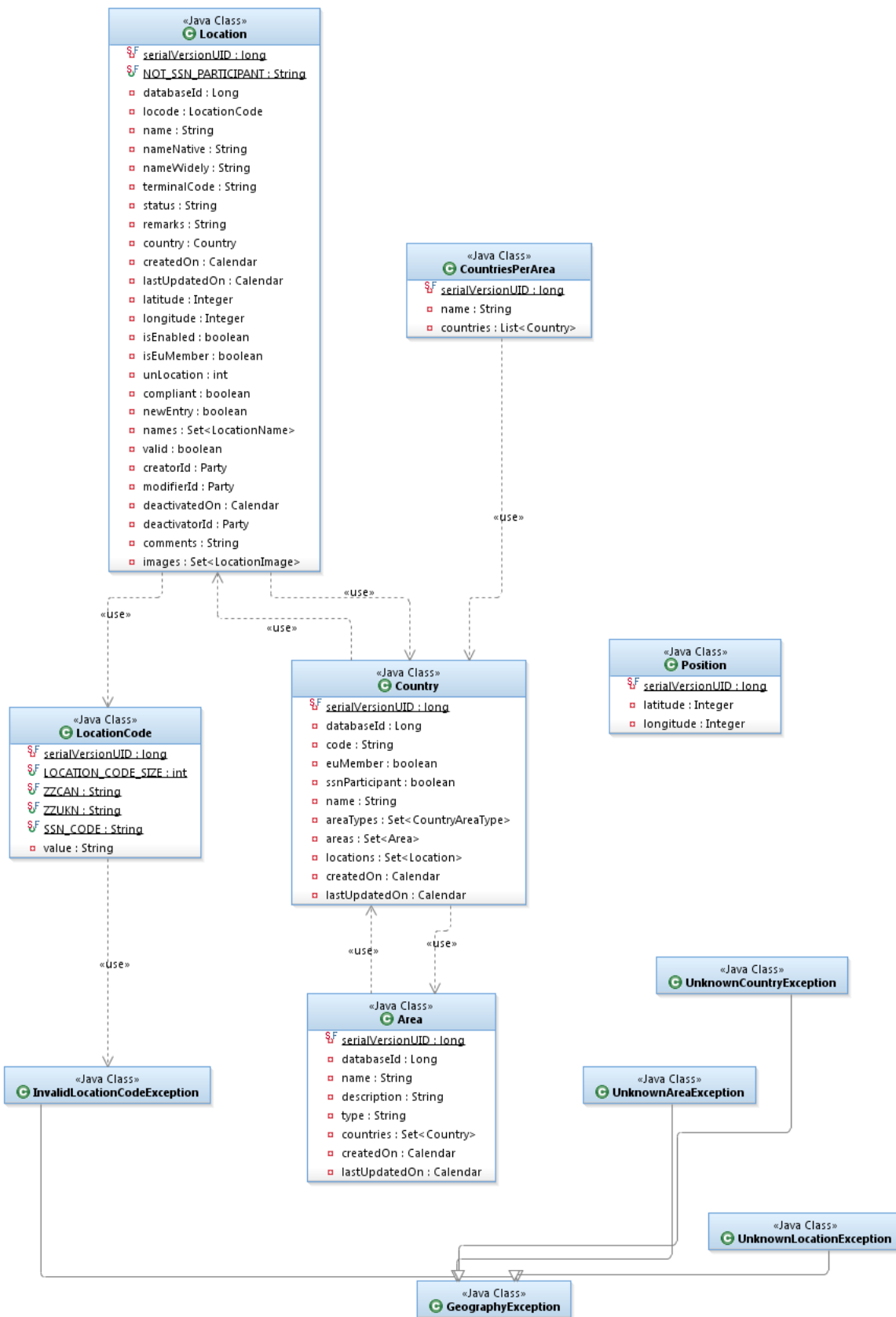
Class Diagram : vessel	
<b>Class</b>	<p><b>Imo</b></p> <p>This class represents an IMO number.</p> <p>EU Member states are expected to send notifications to SSN, using either the IMO or the MMSI in order to identify the vessel</p>
<b>Class</b>	<p><b>Mmsi</b></p> <p>This class represents the MMSI number of a vessel. The MMSI number is a secondary vessel identifier.</p>
<b>Class</b>	<p><b>MaritimeIdentificationDigit</b></p> <p>This class represents a maritime id of a vessel signified by the three first digits of the MMSI number.</p>
<b>Class</b>	<p><b>Vessel</b></p> <p>This class represents the vessel identified by the internal system id and uniquely defined by the IMO number. It holds the vessel attributes such as Banned, Single Hull Tanker, type of ship, year of the construction of the ship, etc.</p>
<b>Class</b>	<p><b>VesselDetail</b></p> <p>This class represents the vessel identification details such as MMSI, CallSign, Name and the vessel Flag; the activated indicator showing whether the vessel is sea worthy, the status vessel validation check (valid, non-valid, temporary), etc.</p>
<b>Class</b>	<p><b>VesselDetailVersion</b></p> <p>This class represents the versions of the vessel definitions in vessels history.</p>
<b>Class</b>	<p><b>VesselDetailHistory</b></p> <p>This class represents the update details of vessel particulars such as the reason and timestamp the updates became effective</p>
<b>Class</b>	<p><b>VesselSearchCriteria</b></p> <p>This class represents the vessel identifiers used as search criteria: IMO number, MMSI number, CallSign and ship Name patterns.</p>
<b>Class</b>	<p><b>VesselVVDetail</b></p> <p>This class represents vessel details used for the vessels validation and verification procedure provided by the management console</p>
<b>Class</b>	<p><b>UploadShtVessels</b></p> <p>A class represents the uploaded single hull tanker used by the upload "Single Hull Tankers" procedure.</p>
<b>Class</b>	<p><b>CannotUpdateTestVesselException</b></p> <p>This exception is thrown by processNotification method in order to indicate that an undefined version of the test vessel is included in a notification message.</p>

Class Diagram : vessel	
<b>Class</b>	<b>CannotUpdateBannedVesselException</b> This exception is thrown by processNotification method in order to indicate that a new version of a banned vessel is included in a notification message.
<b>Class</b>	<b>InvalidVesselException</b> This exception is thrown in order to indicate that a specific reported vessel is invalid.
<b>Class</b>	<b>UnregisteredVesselException</b> This exception is thrown in order to indicate that a specific vessel cannot be found in the SSN vessel registry.

*4.2.2.1.7 Package: geography*

**Class Diagram : geography**

## Class Diagram : geography

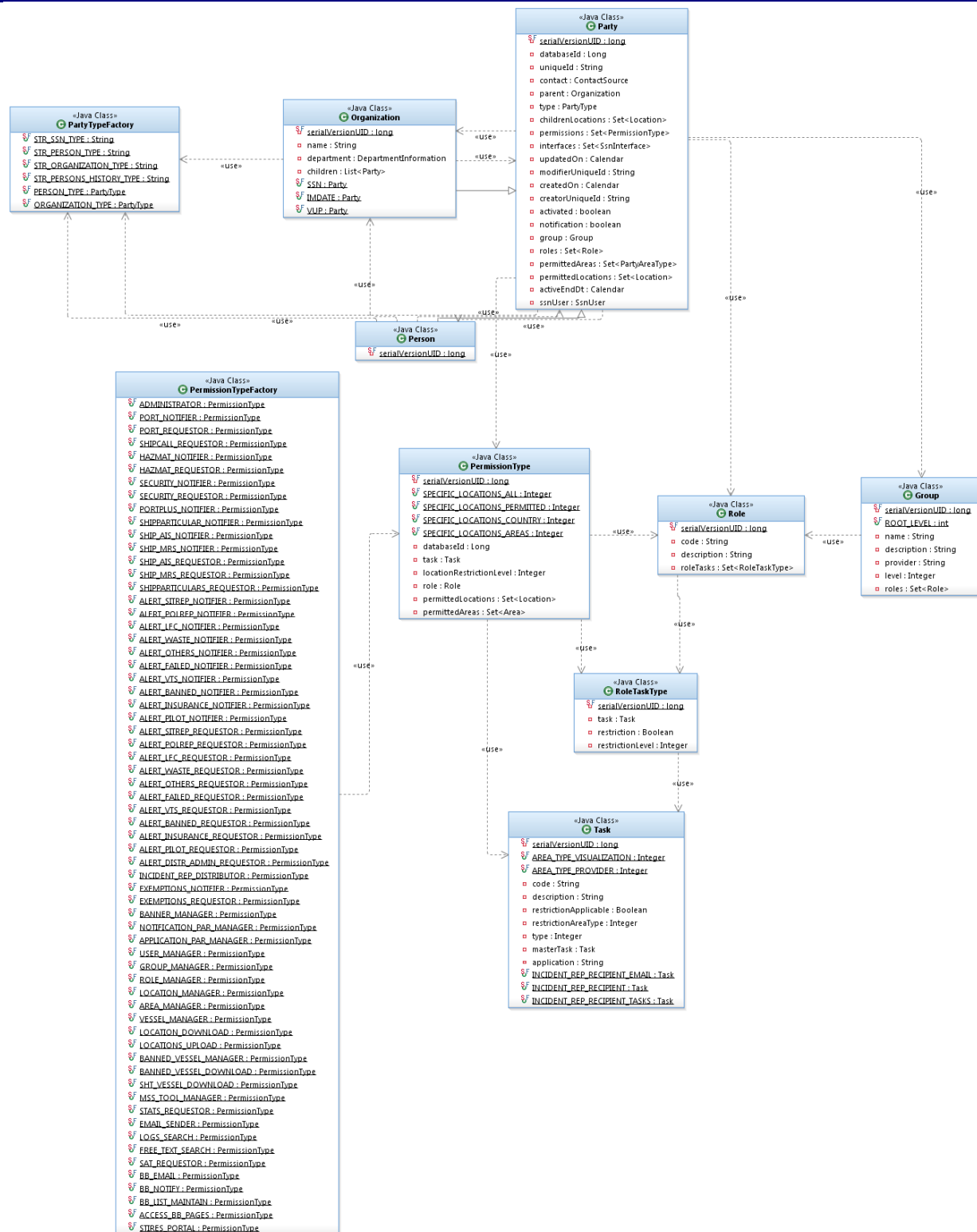


Class Diagram : geography	
<b>Class</b>	<b>Area</b> Representation of the geographical areas.
<b>Class</b>	<b>Country</b> Representation of the countries (EU and NON-EU) defined in SafeSeaNet.
<b>Class</b>	<b>Location</b> Representation of the location codes defined in SafeSeanet. Types of location codes are: SSN Specific, UN Specific and Temporary (marked for validation).
<b>Class</b>	<b>LocationCode</b> Representation of the actual LOCODE for the location codes in SafeSeaNet. LOCODEs may originate from UN or be SSN specific (e.g. for way points).
<b>Class</b>	<b>CountriesPerArea</b> Represents the List of countries that belong to a specific area.
<b>Class</b>	<b>Position</b> Represents the position (longitude, latitude).
<b>Class</b>	<b>UnknownLocationException</b> Is thrown if the specified Location is unknown.
<b>Class</b>	<b>InvalidLocationCodeException</b> Is thrown if the specified LocationCode is not valid.
<b>Class</b>	<b>UnknownAreaException</b> Is thrown if the specified Area is unknown.
<b>Class</b>	<b>UnknownCountryException</b> Is thrown if the specified Country is unknown.

#### 4.2.2.1.8 Package: organisation

#### Class Diagram : organisation

## Class Diagram : organisation



### Class

### Party

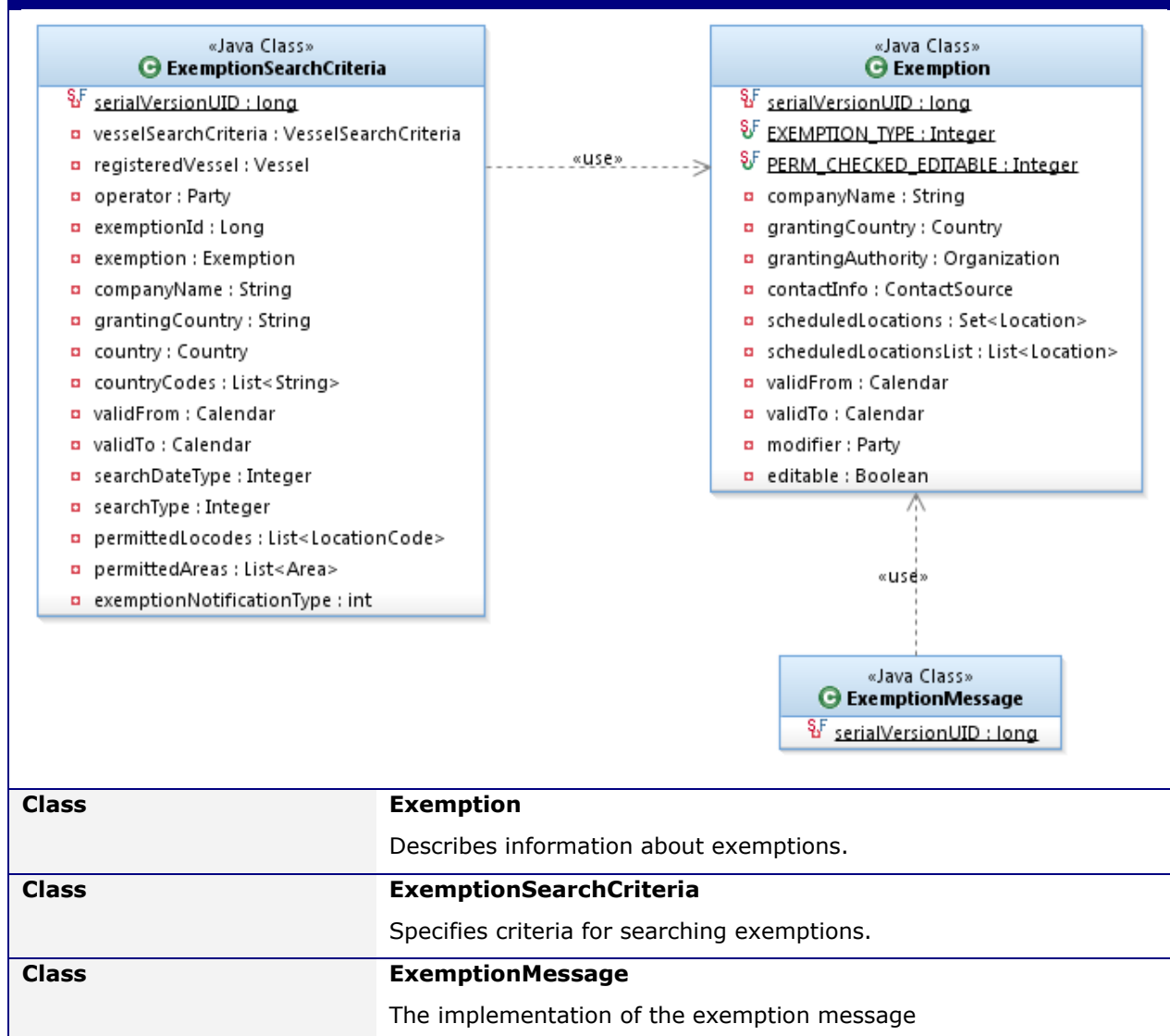
Abstract representation of the Parties (Users and Authorities) in the SSN eco system.

Class Diagram : organisation	
<b>Class</b>	<b>Organisation</b> Representation of the Organisations (Authorities) in the SSN eco system. A new attribute is added to identify the type of the requestor URL receives InformationReplies made by SSN; two types shall be supported: XML (current version) and SOAP (Web Service of Data requestor).
<b>Class</b>	<b>Person</b> Representation of the Persons (Users) in the SSN eco system.
<b>Class</b>	<b>PartyType</b> Representation of the Party types (Users and Authorities)
<b>Class</b>	<b>PartyTypeFactory</b> Create the party types in SafeSeaNet (Users and/or Authorities).
<b>Class</b>	<b>Task</b> Representation of the task defined per function in SafeSeaNet (Send Port Notifications etc).
<b>Class</b>	<b>PermissionType</b> Representation of the Permission types (SendPortNotifications etc) could be assigned to a party
<b>Class</b>	<b>PermissionTypeFactory</b> Create the permission defined per function in SafeSeaNet (Send Port Notifications etc).
<b>Class</b>	<b>Role</b> Representation of the roles (NCA, POR, etc).
<b>Class</b>	<b>Group</b> Representation of the group consisting of a set of roles.
<b>Class</b>	<b>RoleTaskType</b> Class that maps roles to tasks.



#### 4.2.2.1.9 Package: exemption

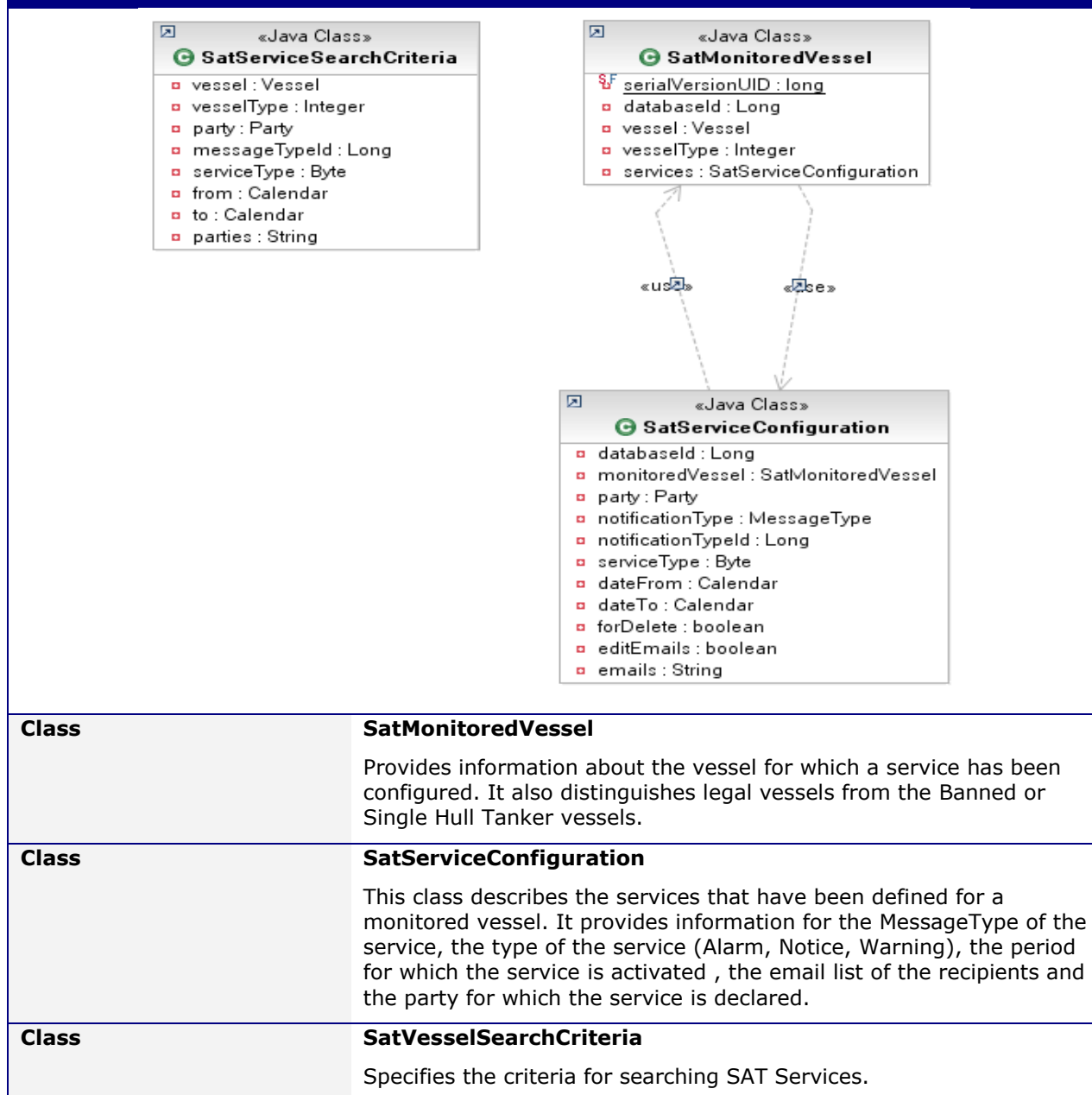
##### Class Diagram : exemption



#### 4.2.2.1.10 Package: sat (Ship Activity Tracking)

##### Class Diagram : sat

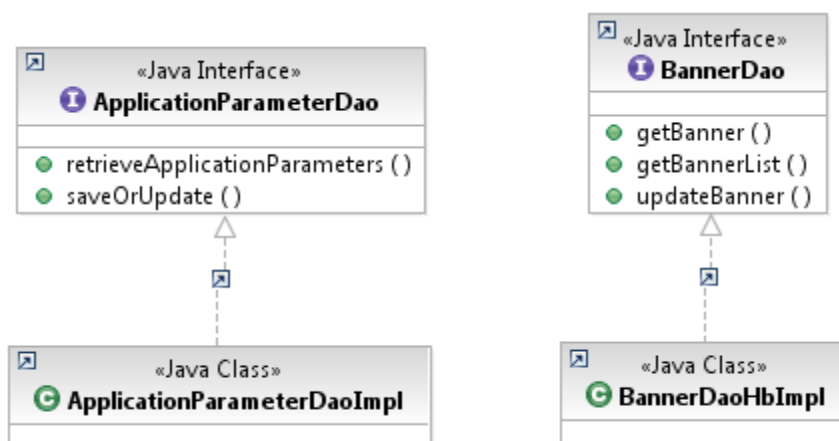
## Class Diagram : sat



#### 4.2.2.2 Module ssn-dao

##### 4.2.2.2.1 Package: common

**Class Diagram : common-dao**



<b>Interface</b>	<b>ApplicationParameterDao</b> A data access object for managing the database operations for the application parameters
<b>Class</b>	<b>ApplicationParameterImpl</b> The implementation of the ApplicationParameterDao.
<b>Interface</b>	<b>BannerDao</b> A data access object for managing the database operations for the Ticket Banner text displayed on the Web console header.
<b>Class</b>	<b>BannerDaoHbImpl</b> Hibernate based implementation of the BannerDao.

##### 4.2.2.2.2 Package: message-log-dao

**Class Diagram : message-log-dao**

## Class Diagram : message-log-dao



<b>Interface</b>	<b><u>MessageLogItemDao</u></b> Database data access object for the management of log items.
<b>Class</b>	<b><u>MessageLogItemDaoHbImpl</u></b> Implements the MessageLogItemDao interface
<b>Interface</b>	<b><u>StatisticsDao</u></b> Database data access object for the statistics functionality.

## Class Diagram : message-log-dao

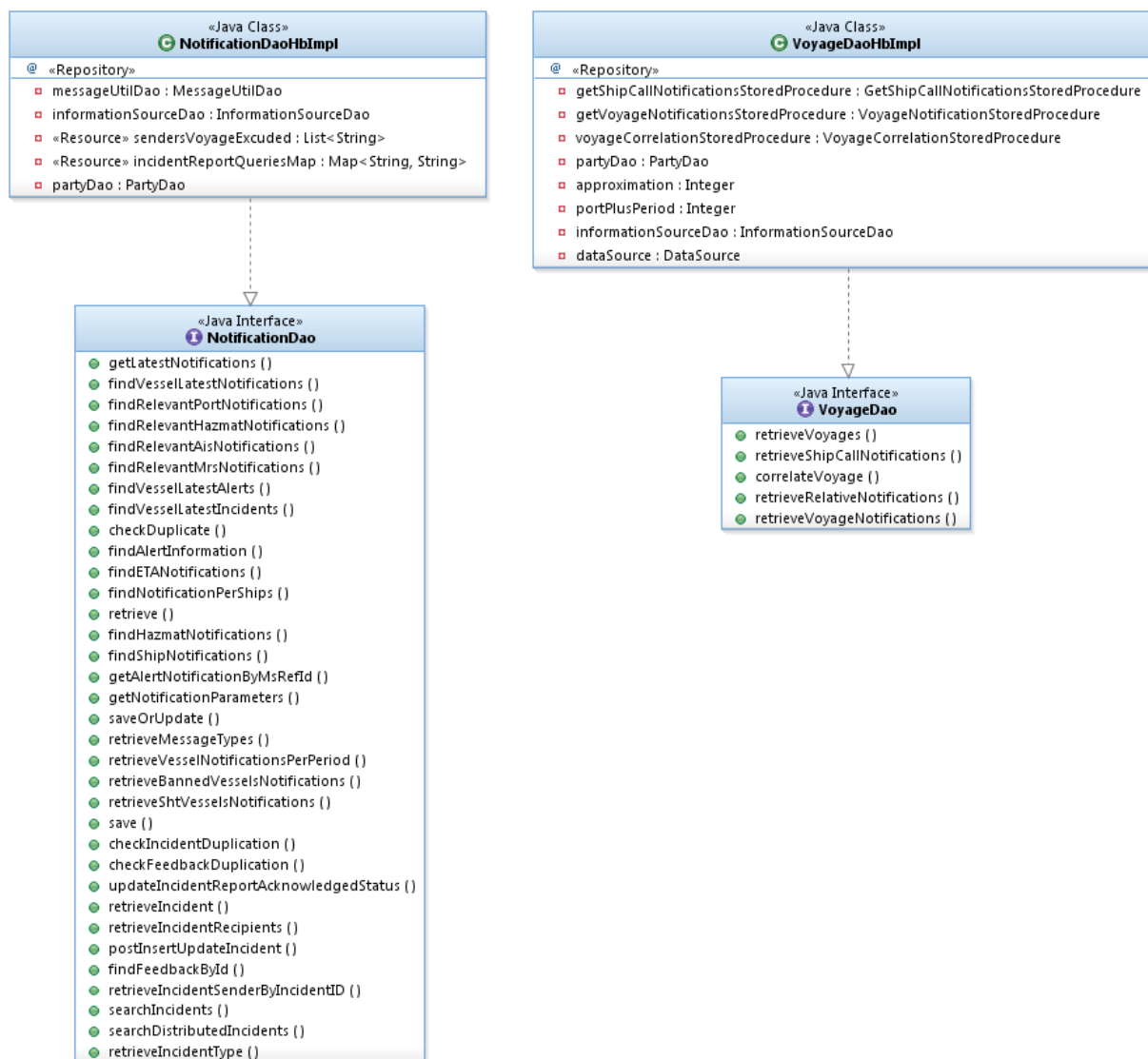
### Class

### StatisticsDaoJdbcImpl

Implements the StatisticsDao interface

## 4.2.2.2.3 Package: notification-dao

## Class Diagram : notification-dao



### Interface

### **NotificationDao**

A data access object for managing the database operations for notifications.

### Class

### **NotificationHbImpl**

Hibernate based implementation of the NotificationDao.

### Interface

### **VoyageDao**

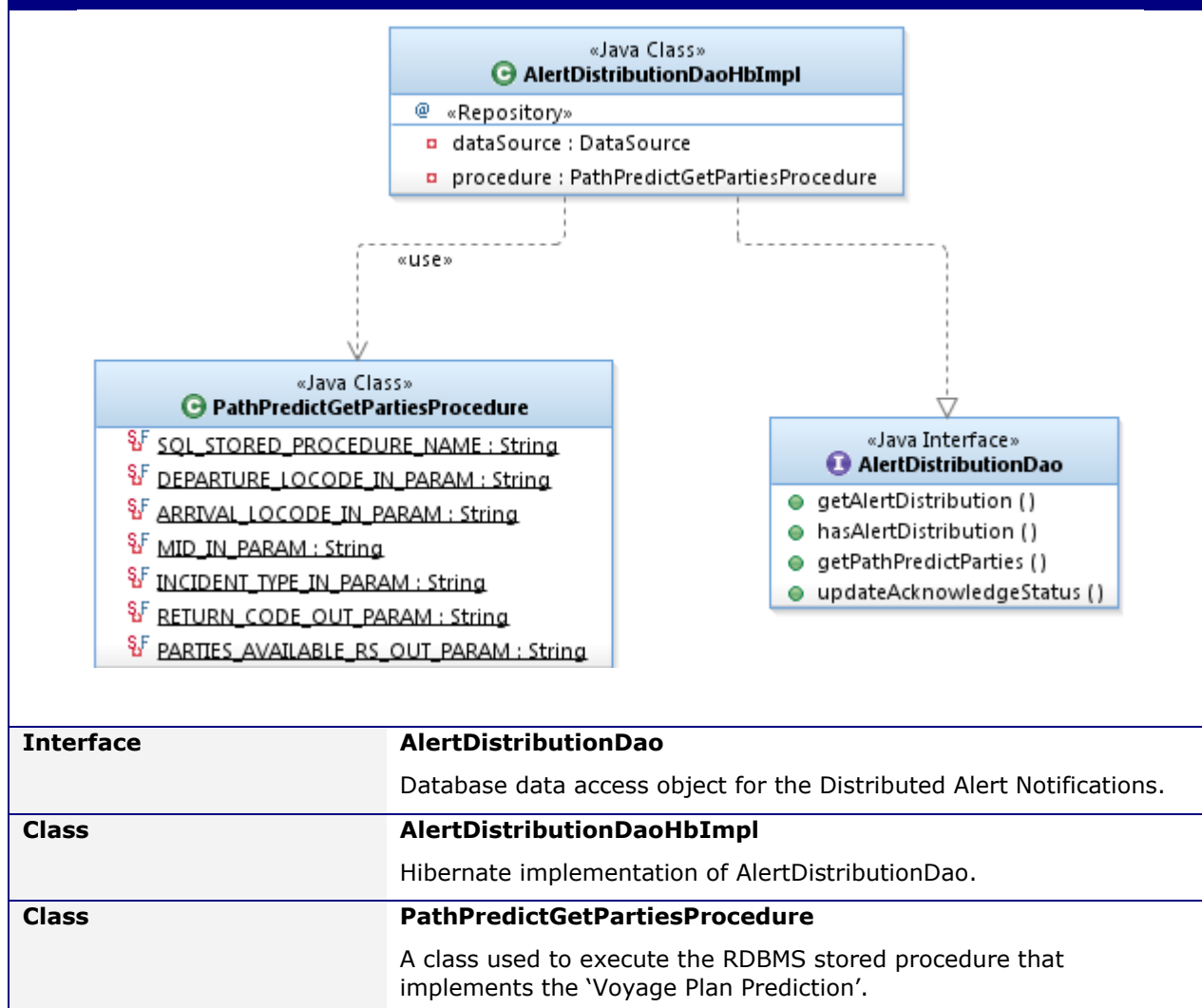
A data access object for managing the database operations for voyages.

#### Class Diagram : notification-dao

<b>Class</b>	<b>VoyageHbImpl</b> Hibernate based implementation of the VoyageDao.
--------------	---

#### 4.2.2.2.4 Package: alert-distribution-dao

#### Class Diagram : Alert Distribution Dao package



#### 4.2.2.2.5 Package: geography-dao

#### Class Diagram : geography-dao

## Class Diagram : geography-dao



Interface

GeographyDao

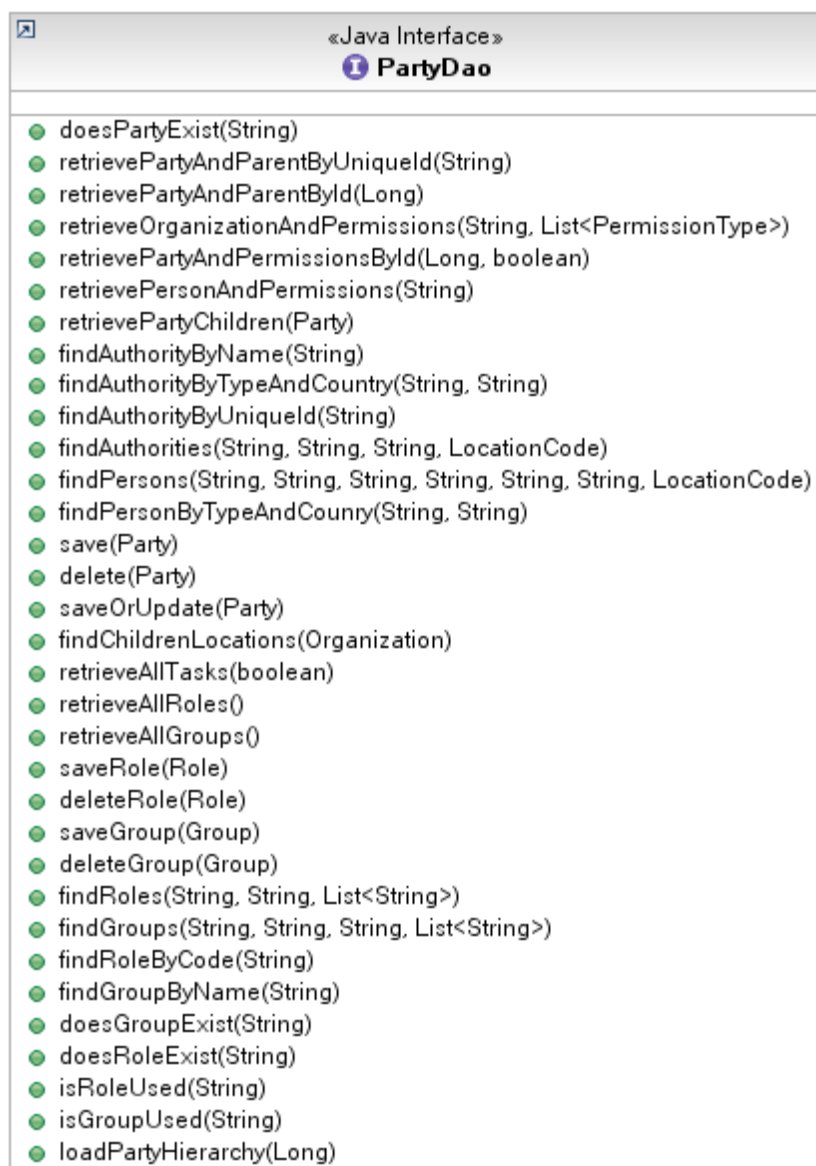
#### Class Diagram : geography-dao

	A data access object for managing the database operations for Locations.
<b>Class</b>	<b>GeographyHbImpl</b> Hibernate based implementation of the GeographyDao.
<b>Class</b>	<b>LoadLocodeProcedure</b> A class used to execute the RDBMS stored procedure that implements the 'LOCODE Management from UNECE' procedure.
<b>Interface</b>	<b>UploadDao</b> A data access object for managing the CSV file stored on EIS database.
<b>Class</b>	<b>UploadJdbcImpl</b> JDBC based implementation of the UploadDao.

#### Class Diagram : organization-dao



## Class Diagram : organization-dao



### Interface

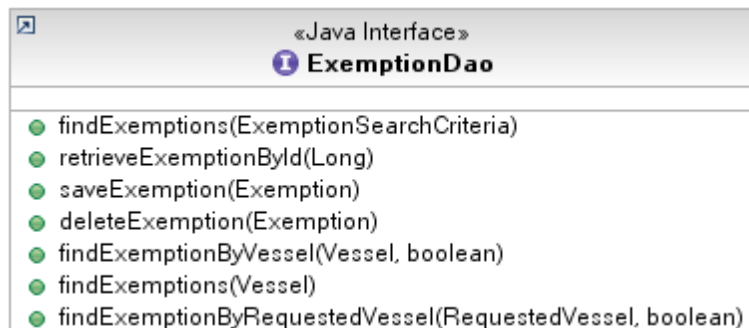
### PartyDao

A data access object for managing the database operations for parties and their permissions.

#### 4.2.2.2.6 Package: exemption-dao

## Class Diagram : exemption-dao

#### Class Diagram : exemption-dao



#### Interface

#### ExemptionDao

Interface describing the data access object of Exemptions.

#### 4.2.2.2.7 Package: sat-dao

#### Class Diagram : sat-dao



#### Interface

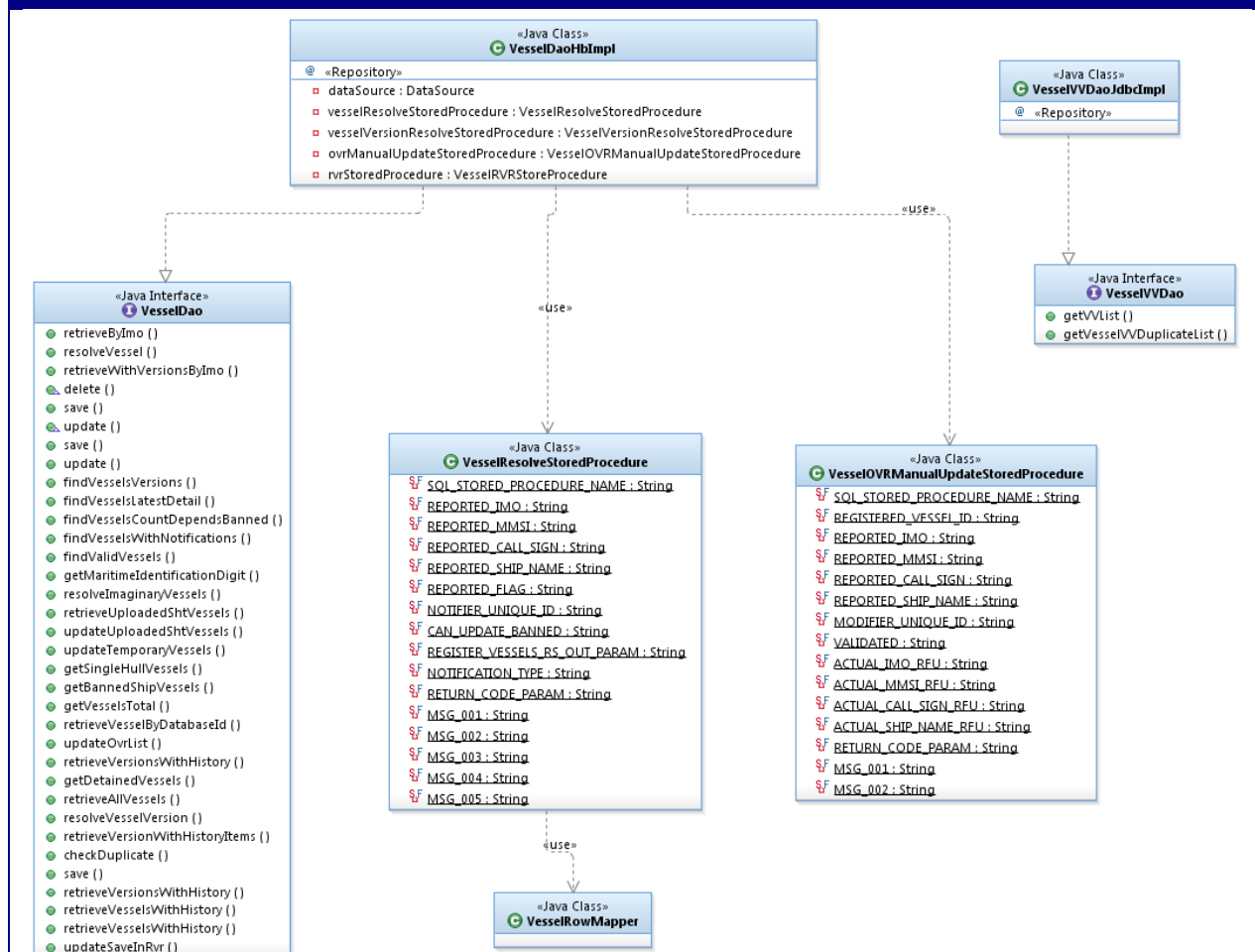
#### ShipActivityTrackingDao

Interface describing the data access object for sat service configurations.

#### 4.2.2.2.8 Package: vessel-dao

#### Class Diagram : vessel-dao

## Class Diagram : vessel-dao

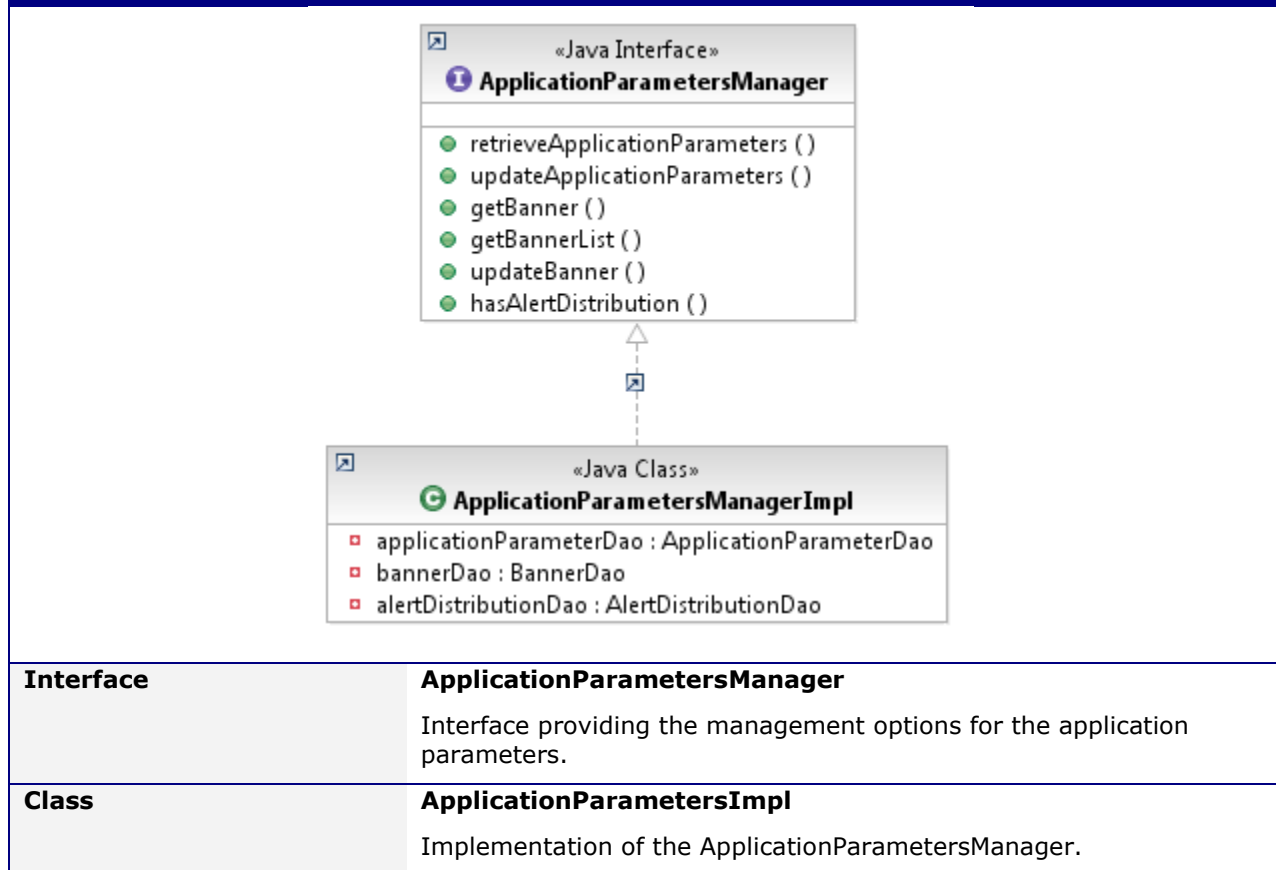


<b>Interface</b>	<b>VesselDao</b> A data access object for managing the database operations for vessels
<b>Class</b>	<b>VesselDaoHbImpl</b> Hibernate based implementation of the VesselDao.
<b>Class</b>	<b>VesselResolveStoredProcedure</b> A class used to execute the RDBMS stored procedure that implements 'the vessel's resolve on notification arrival'
<b>Class</b>	<b>VesselOVRManualUpdateStoredProcedure</b> A class used to execute the RDBMS stored procedure that implements the 'OVR manual update' that is part of the vessels validation and verification procedure.
<b>Interface</b>	<b>VesselIVVDao</b> A data access object for managing the database operations for vessels validation and verification procedure.
<b>Class</b>	<b>VesselIVVDaoJdbcImpl</b> JDBC based implementation of the VesselIVVDao.

#### 4.2.2.3 Module: ssn-core

##### 4.2.2.3.1 Package: common-manager

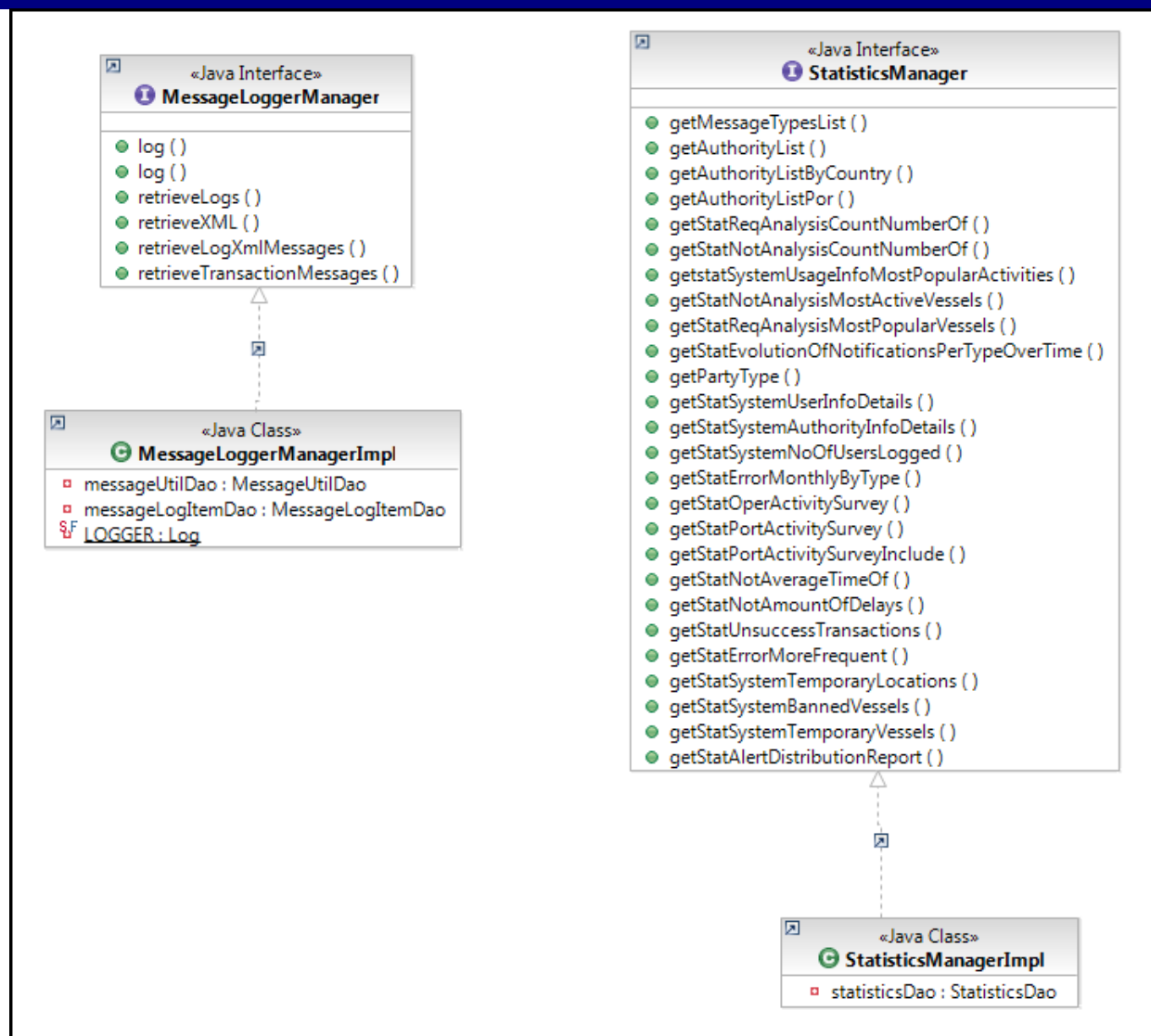
###### Class Diagram : common-manager



##### 4.2.2.3.2 Package: message-log-manager

###### Class Diagram : message-log-manager

### Class Diagram : message-log-manager

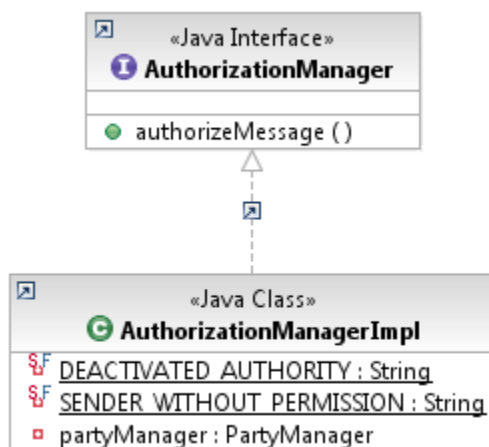


<b>Interface</b>	<b><u>MessageLoggerManager</u></b> Interface providing the management options for MessageLogItem.
<b>Class</b>	<b><u>MessageLoggerManagerImpl</u></b> Implements the management of log items; it creates/saves and searches the ssn logs.
<b>Interface</b>	<b><u>StatisticsManager</u></b> Interface providing the statistics functionality.
<b>Class</b>	<b><u>StatisticsManagerImpl</u></b> Implements the statistics functionality.

#### 4.2.2.3.3 Package: message-manager

### Class Diagram : message-manager

## Class Diagram : message-manager



### Interface **AuthorizationManager**

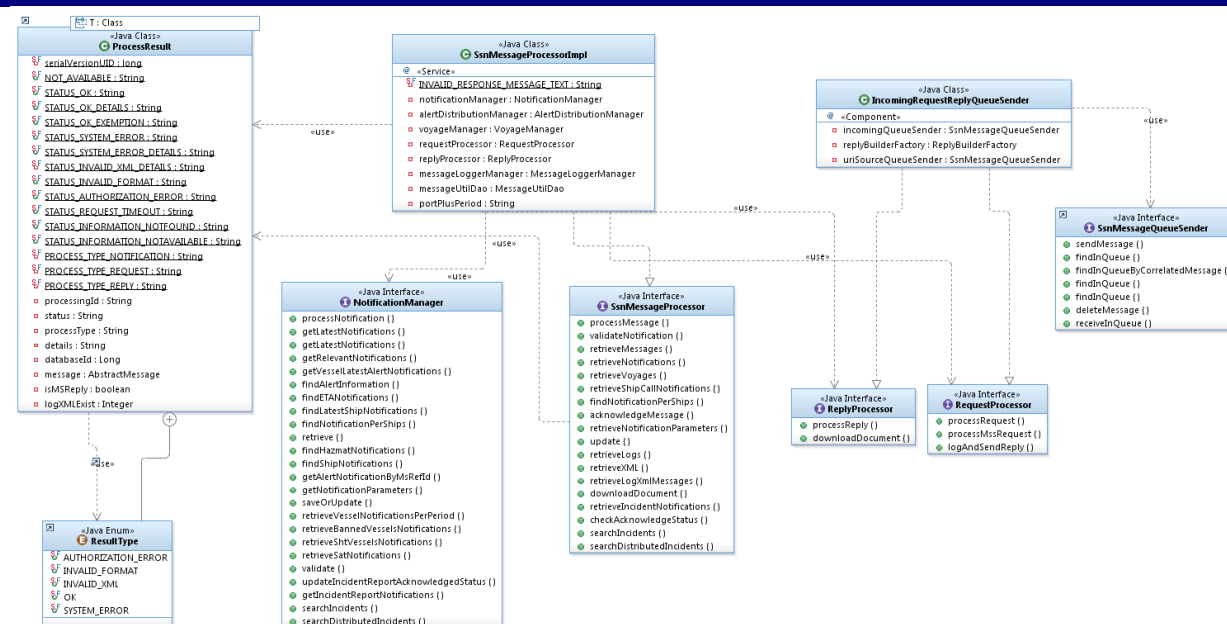
An interface provides the authorization of the incoming xml messages.

### Class **AuthorizationManagerImpl**

Implements the authorization of the incoming xml messages via/using the Party manager.

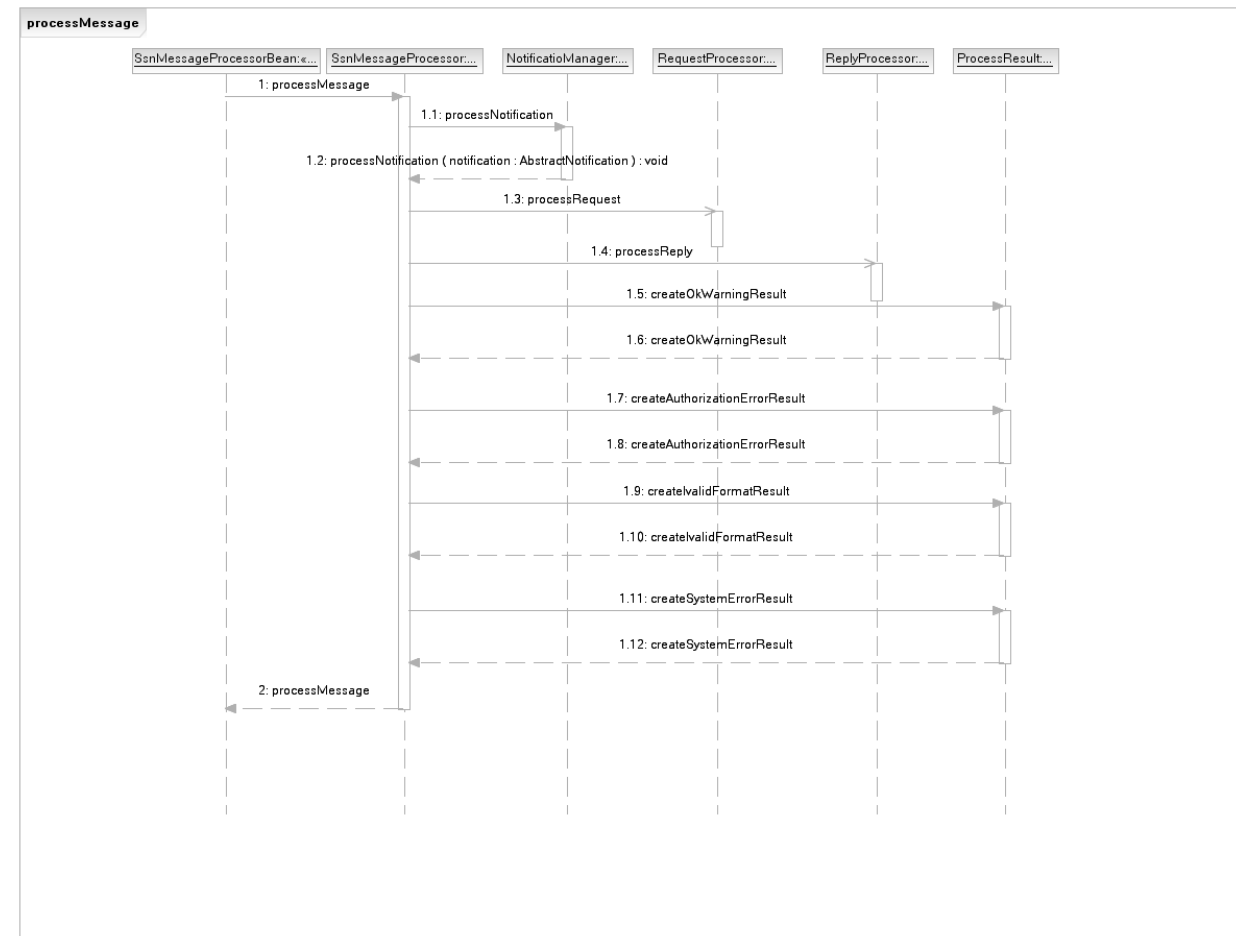
## 4.2.2.3.4 Package: message-processor

## Class Diagram : message-processor



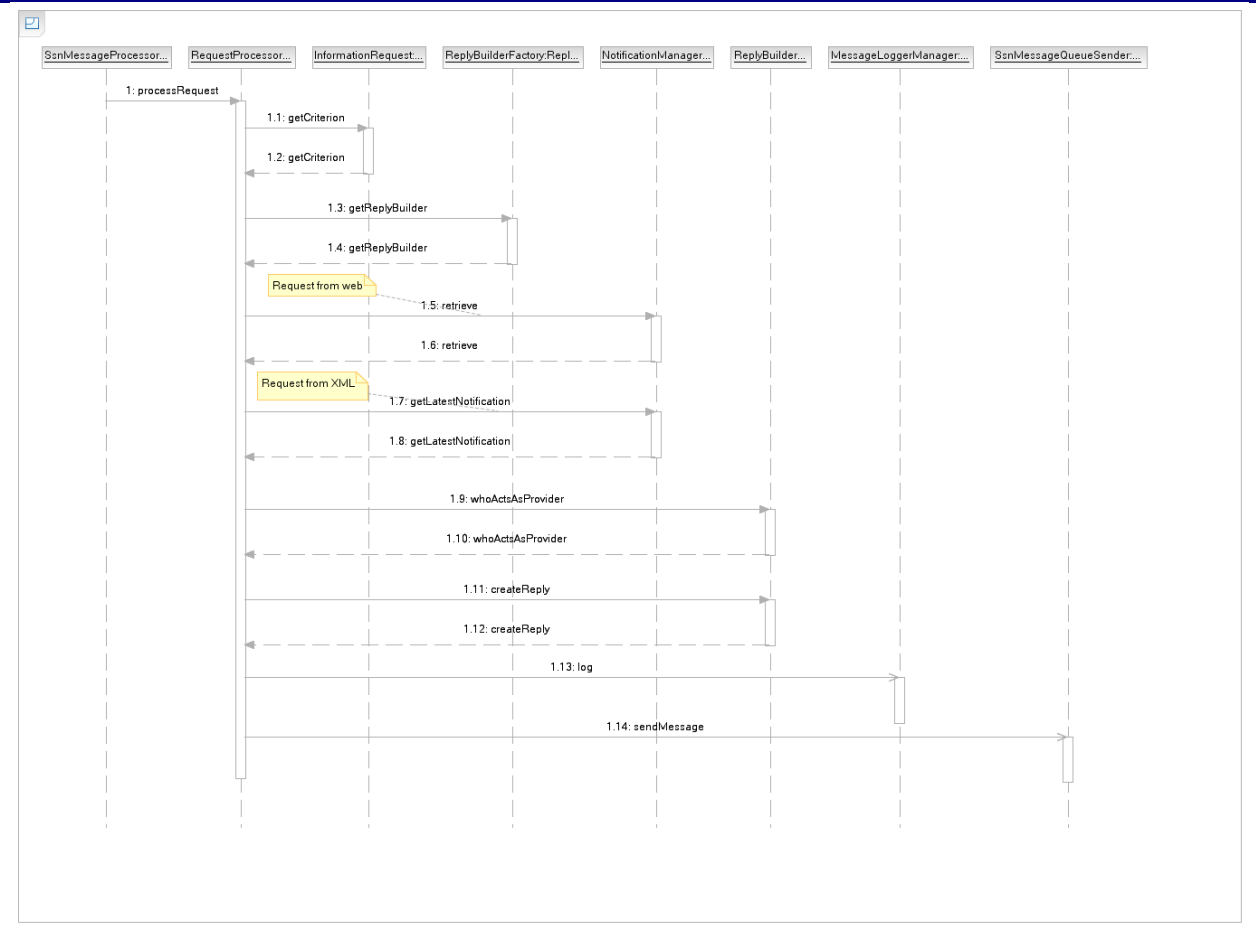
## Sequence Diagram : Process Message- Notification & Request/Reply

## Class Diagram : message-processor



## Sequence Diagram : Process Message- Reply

### Class Diagram : message-processor

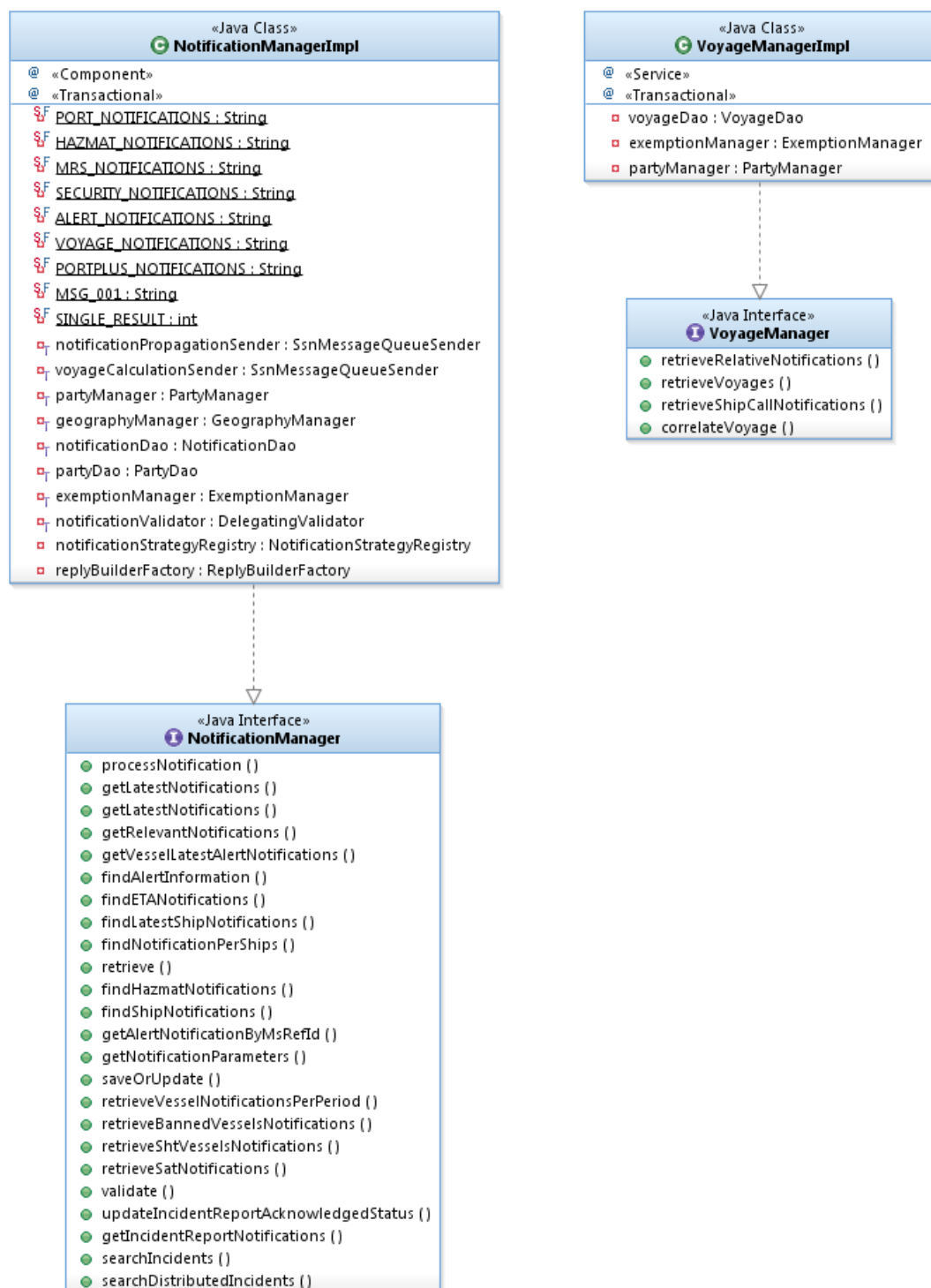


#### 4.2.2.3.5 Package: notification-manager

### Class Diagram : notification-manager



## Class Diagram : notification-manager



### Interface

### NotificationManager

Interface providing the management options for notification registry.

### Class

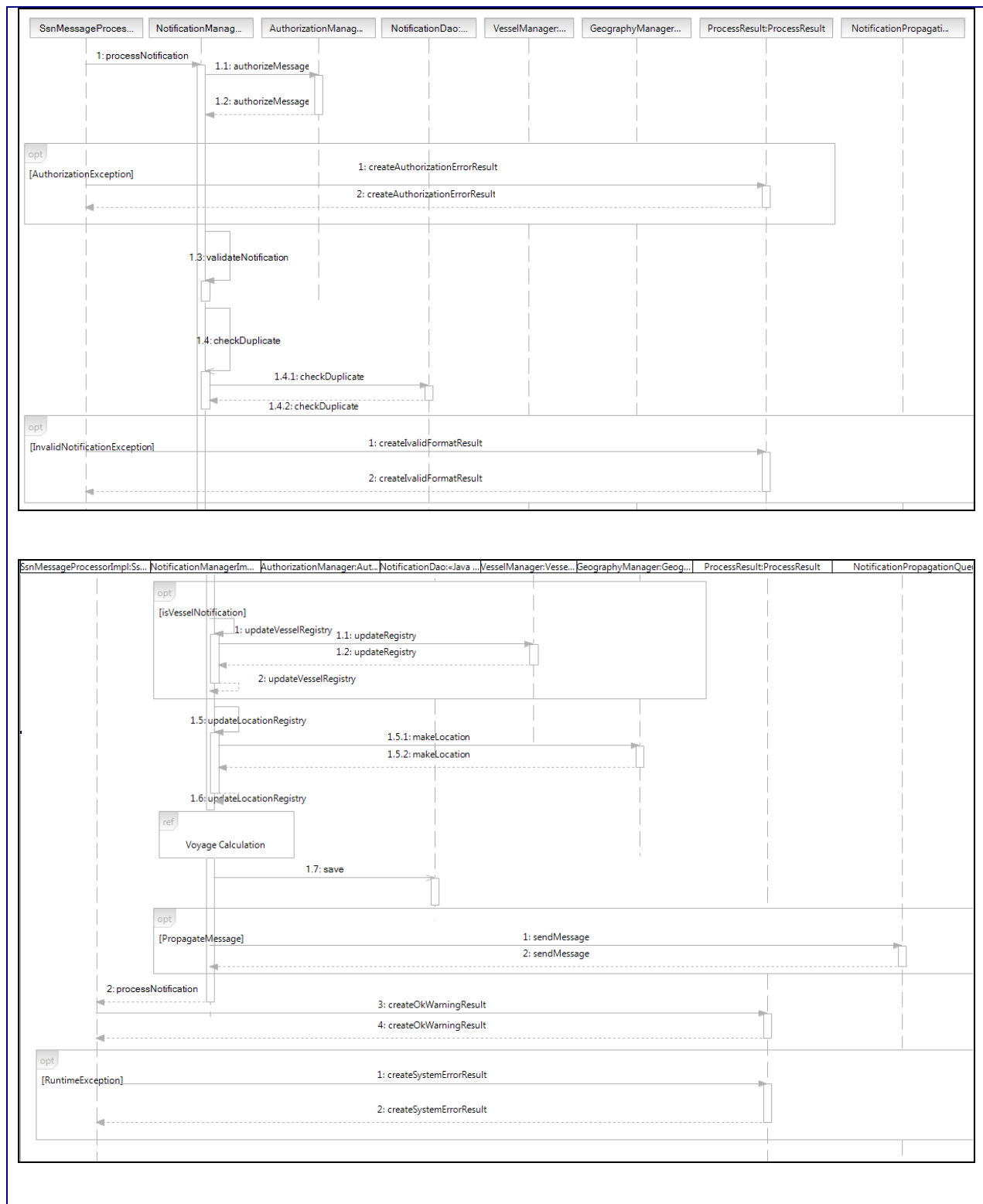
### NotificationManagerImpl

Implementation of the NotificationManager.

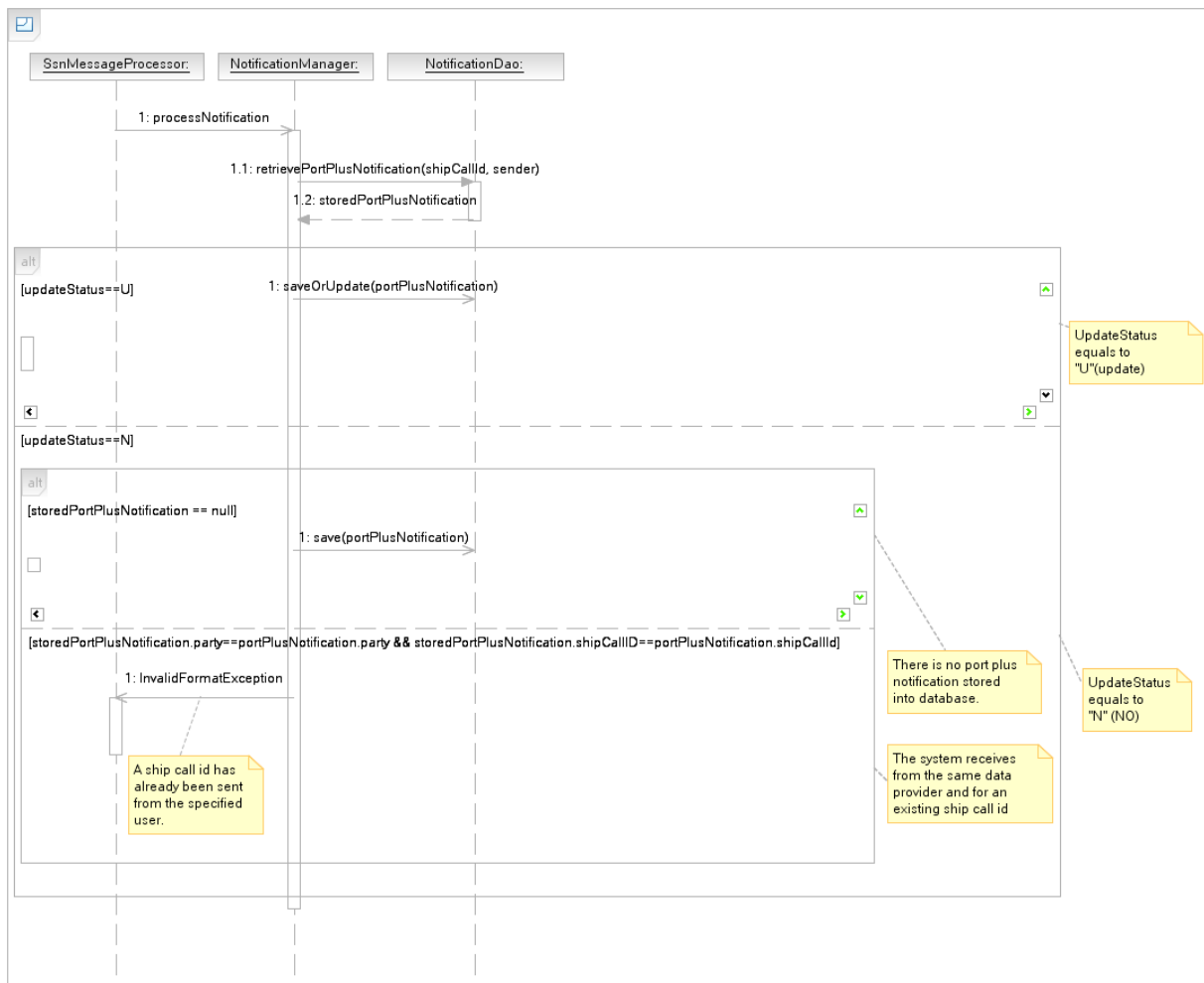
#### Class Diagram : notification-manager

<b>Interface</b>	<b>VoyageManager</b> Interface providing the management options for voyages.
<b>Class</b>	<b>VoyageManagerImpl</b> Implementation of the VoyageManager.

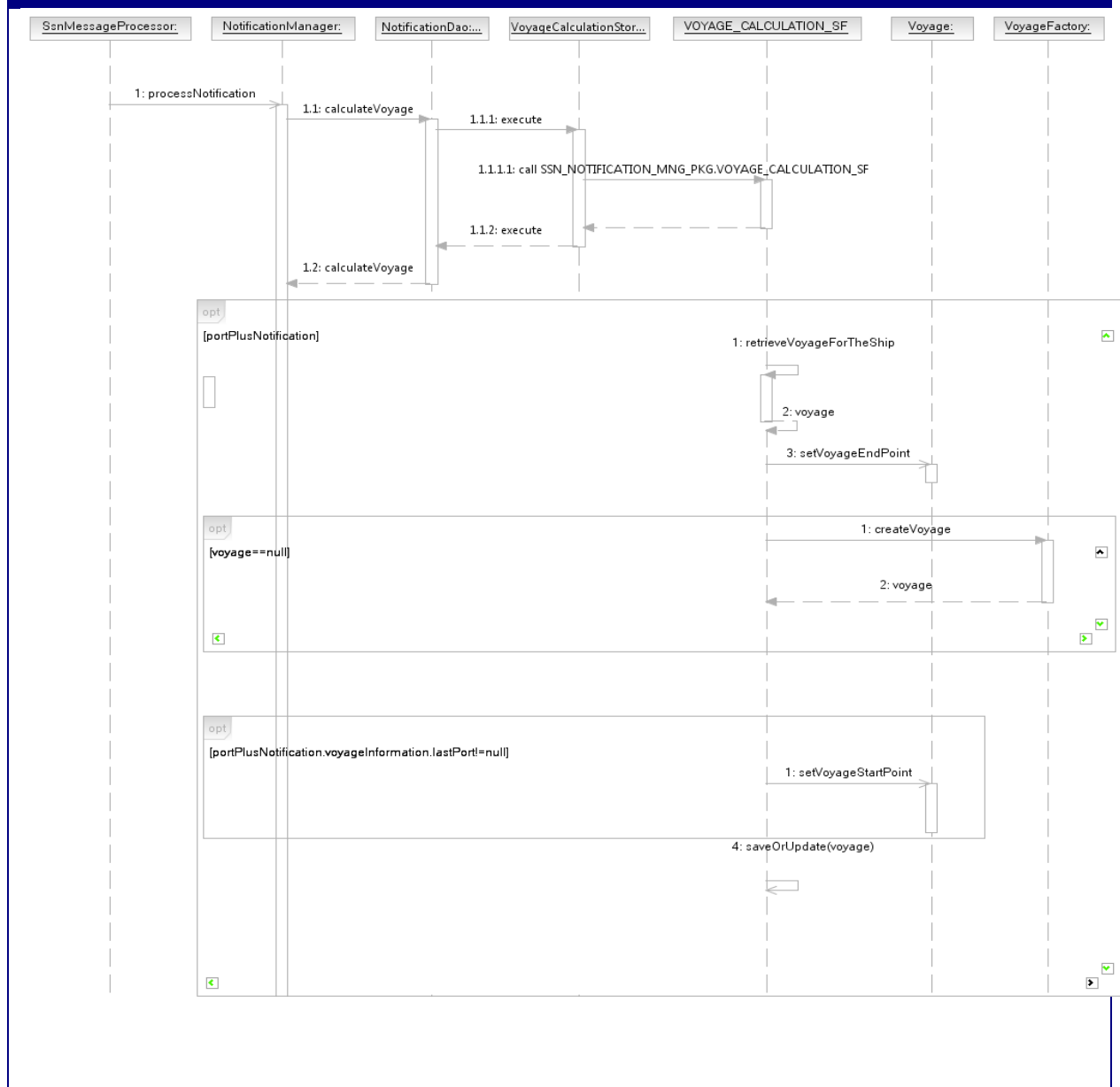
#### Sequence Diagram : Notification Processing - Valid Notification



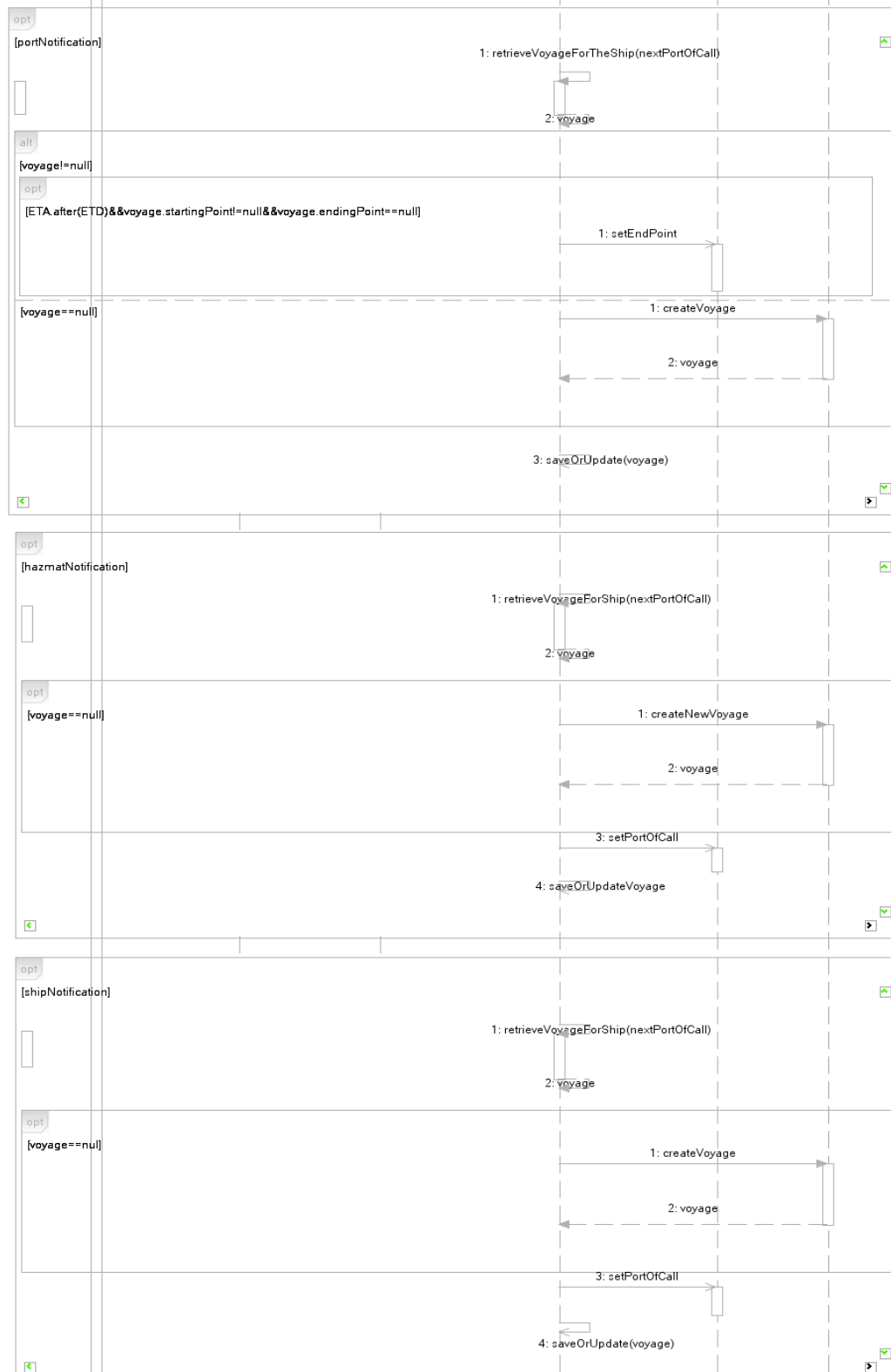
## Sequence Diagram : Notification Processing – PortPlus Notification



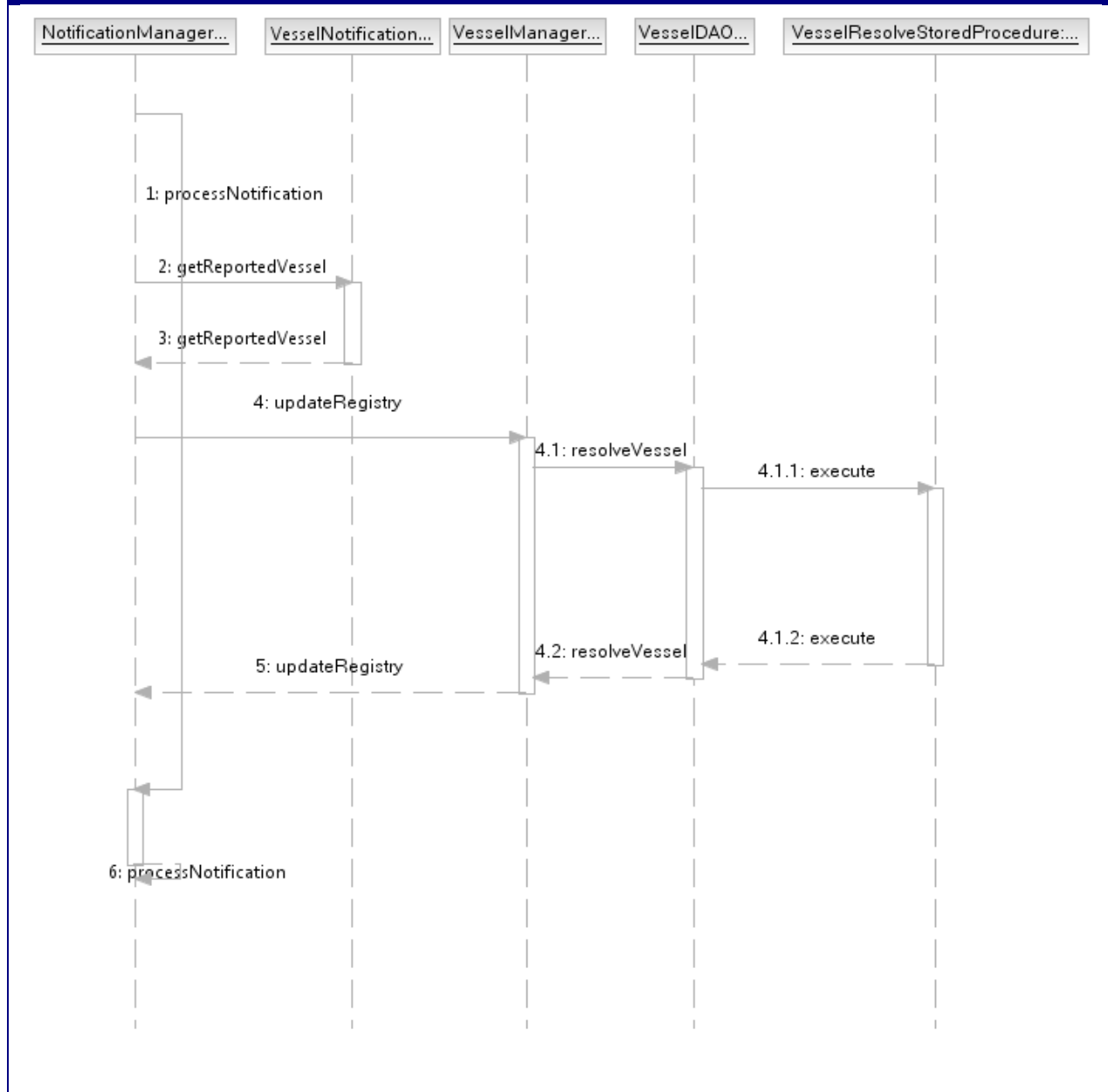
## Sequence Diagram : Notification Processing – Voyage Calculation



## Sequence Diagram : Notification Processing – Voyage Calculation

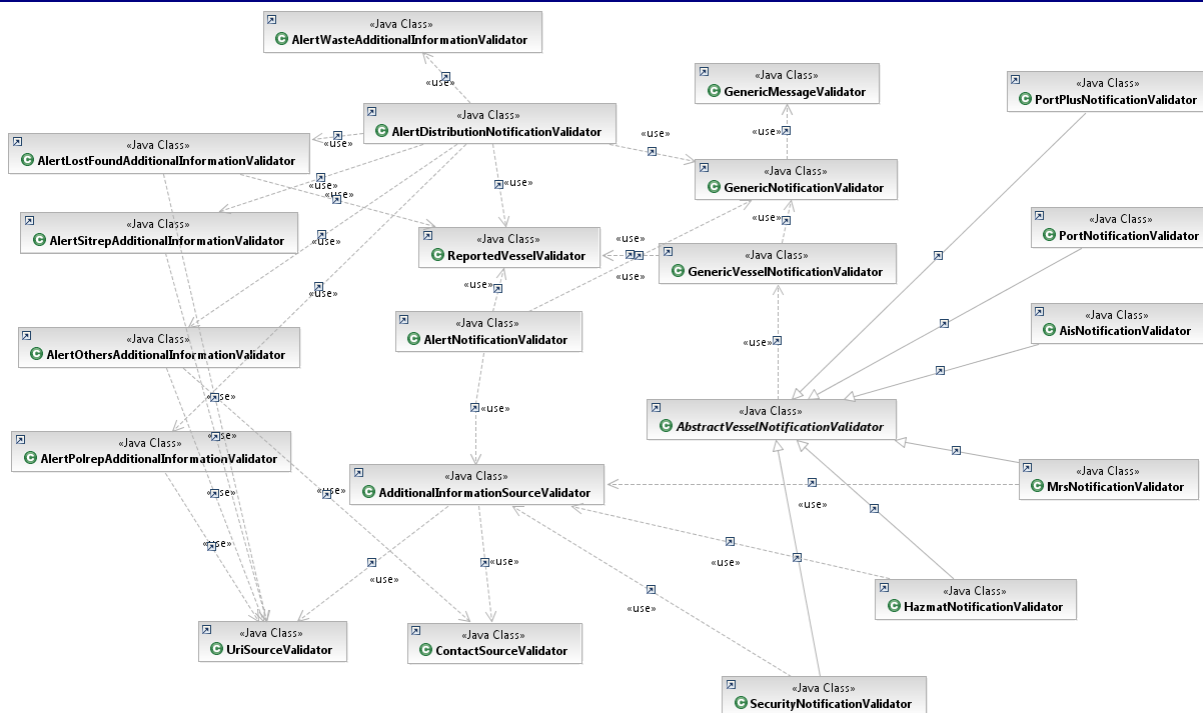


### Sequence Diagram: Update Vessel Registry Using Reported vessel (IMO and MMSI Number, CallSign, Ship name)



### Class Diagram : notification-validation

## Class Diagram : notification-validation



<b>Package</b>	<b>notification.validation</b>
<b>Class</b>	<b>GenericMessageValidator</b> Validates the notification's header – MsRefId, sender, receiver, test case id.
<b>Class</b>	<b>GenericNotificationValidator</b> Validates the generic notification attributes.
<b>Class</b>	<b>GenericVesselNotificationValidator</b> Invokes the reported vessel validator.
<b>Class</b>	<b>ReportedVesselValidator</b> Validates the IMO number, MMSI number, the Call Sign, the Ship Name and Flag of the reported vessel. It actually delegates the validation of the vessel particulars to the corresponding vessel validators.
<b>Class</b>	<b>AbstractVesselNotificationValidator</b> This class is the common parent of all classes that implement discrete notification validators.
<b>Class</b>	<b>AisNotificationValidator</b> The business rules for the AIS (Ship) notification.
<b>Class</b>	<b>AlertNotificationValidator</b> The business rules for the Alert notification for identified and non-identified vessels and the additional information source details. In case of notification for identified vessel, the ReportedVesselValidator is invoked.



Class Diagram : notification-validation	
<b>Class</b>	<b>HazmatNotificationValidator</b> The business rules for the Hazmat notification and the additional information source details
<b>Class</b>	<b>MrsNotificationValidator</b> The business rules for the MRS (Ship) notification and the additional information source details
<b>Class</b>	<b>PortNotificationValidator</b> The business rules for the Port notification.
<b>Class</b>	<b>PortPlusNotificationValidator</b> The business rules for the PortPlus notification.
<b>Class</b>	<b>SecurityNotificationValidator</b> The business rules for the Security notification and the additional information source details
<b>Package</b>	<b>alertdistribution.validation</b>
<b>Class</b>	<b>AlertDistributionNotificationValidator</b> The business rules for the Alert distribution notification. The corresponding validator is invoked according to the notification incident type; e.g. for incident type 'Others' the AlertOthersAdditionalInformationValidator is invoked.
<b>Package</b>	<b>common.validation</b>
<b>Class</b>	<b>AdditionalInformationSourceValidator</b> The business rules for the notification details source (Contact source, URI).
<b>Class</b>	<b>ContactSourceValidator</b> Validates the contact's details such as location, name, email, phone/fax numbers.
<b>Class</b>	<b>UriSourceValidator</b> Validates the URI details

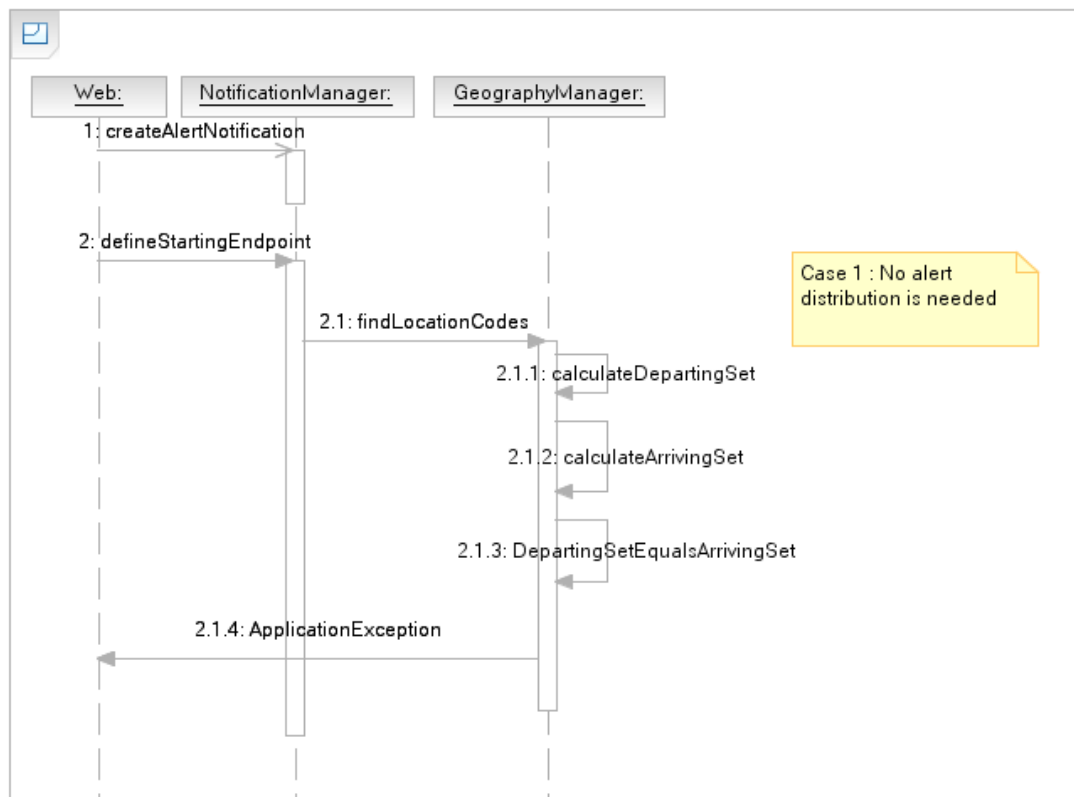
#### 4.2.2.3.6 Voyage Plan Prediction

The process described in the following sequence diagram concerns the prediction of the voyage Plan based on the departing and arriving location code given by the Web user. As specified in the voyage plan prediction algorithm, later in this subsection, three alternative cases exists: cases 1, 2 and 3. Each of the alternative cases is presented in a separate sequence diagram.

A PL/SQL stored procedure implements this algorithm for performance purposes – it avoids JDBC database roundtrips. As an input the package takes the Starting and Ending points (location codes) that the user specifies in the Alert Distribution Web application. As an output the package provides the list of proposed recipients based on the location codes to be produced by the algorithm and the NCA authorities that reside on a location code selected.

## Sequence Diagram : Predict Voyage Plan

Case 1:

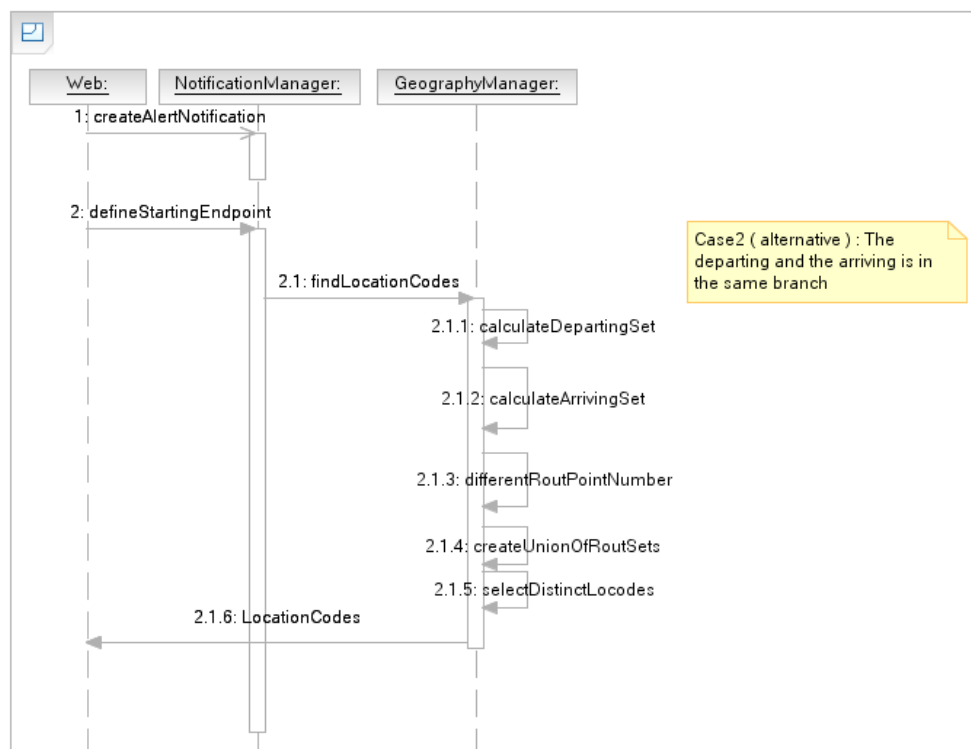


Case 2 (alternative):

## Sequence Diagram : Predict Voyage Plan



Case 3 ( alternative ):



The basic description is presented in the Table 4-5. The prediction of the plan is based primarily on the definition of the Rout-Points. The convention used for defining a Rout-Point is **R ( i, j, k, l )**, where:

- **R**⇒ Stand for Rout-Point
- **i**⇒ Is the Level of the branch. Values: 1 is the basic branch; 2 is the first branch level out of branch 1; level 3 is a branch to branch 2 and so on.
- **j**⇒ Is the branch number. For level 1, the branch number “b” is omitted since only one branch is considered at this level.
- **k**⇒ Is the Rout-Point number for that specific branch.
- **l**⇒ Is an activity parameter. Values: 1 = active (send alert notification); 0 = inactive (do not send alert notification details).

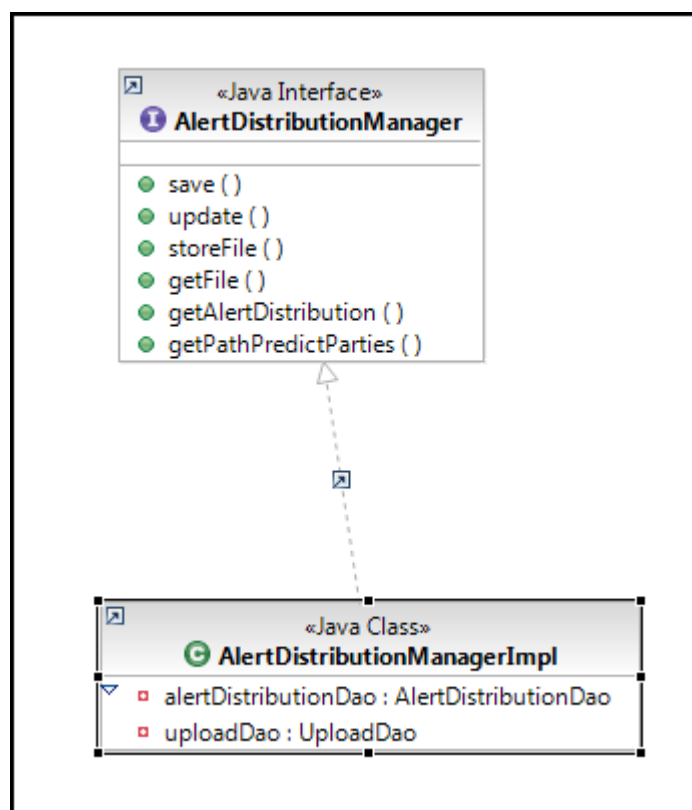
Functional description				
Input →				VoyagePlan( Departing Set; Arriving Set)
1.1				R(i <sub>1</sub> ; j <sub>1</sub> ; k <sub>1</sub> ; l <sub>1</sub> ) = Departing set R(i <sub>2</sub> ; j <sub>2</sub> ; k <sub>2</sub> ; l <sub>2</sub> ) = Arriving set
	2.1			If i <sub>1</sub> = i <sub>2</sub> and j <sub>1</sub> = j <sub>2</sub> and k <sub>1</sub> = k <sub>2</sub> then Case 1
	2.2			# The departing and arriving is in the same branch and set
		3.1		Case 1
		3.2		Message: The departure and the arriving are in the same area. No Alert distribution is needed
	2.3			If i <sub>1</sub> = i <sub>2</sub> and j <sub>1</sub> = j <sub>2</sub> and k <sub>1</sub> ≠ k <sub>2</sub> then Case 2
	2.4			# The departing and arriving is in the same branch
		3.3		Case 2
		3.4		n = 0
		3.5		a = 0
		3.6		m <sub>1</sub> = k <sub>2</sub> - k <sub>1</sub>
		3.7		m <sub>2</sub> =  m <sub>1</sub>  /m <sub>1</sub>
		3.8		n <sub>1</sub> = 0 * m <sub>2</sub> (the increment)
		3.9		# Create the Array A <sub>1</sub> (i <sub>1</sub> +n; j <sub>1</sub> +n; k <sub>1</sub> + n <sub>1</sub> ; l <sub>1</sub> )
		3.10		A <sub>a</sub> (i <sub>1</sub> +n; j <sub>1</sub> +n; k <sub>1</sub> + n <sub>1</sub> ; l <sub>1</sub> )= R(i <sub>1</sub> +n; j <sub>1</sub> +n; k <sub>1</sub> + n <sub>1</sub> ; l <sub>1</sub> )
		3.11		n <sub>1</sub> = n <sub>1</sub> + m <sub>2</sub>
		3.12		a = a + 1
		3.13		If  n <sub>1</sub>   >  m <sub>1</sub>   then exit else go to 3.10
			4.1	Create a Union of the sets in A <sub>a</sub> with single occurrence of countries, and the NCA's shall be identified
		3.14		Exit Case 2
	2.5			# The departing port and the arriving port are in different branches, level 1

			or 2
	2.6		If $i_1 = i_2 = 2$ and $j_1 \neq j_2$ then Case 3
		3.8	Case 3
		3.9	# For simplified reasons we calculate this occurrence as above. Each branch is connected together, so we can use the method above; just anticipate that we have 2 or three separate branches. From start Set to Set before intersection ( $k=1$ ), Then find intersection with the main branch, Set $R(1.1.k_1.l)$ . Then find the arriving Set in branch 1, Set $R(1.1.k_2.l)$ . For this case use method under 2.4 for each part. The total Array will be: $A_T = A_1 \cap A_2$ . Recognise that an Array consists of one or more Sets, $A = R(i_1; j_1; k_1; l_1) \cap R(i_2; j_2; k_2; l_2) \cap \dots \cap R(i_n; j_n; k_n; l_n)$ . If we have the arriving port in an other level two we will then have 3 branches, then we will get the following result: $A_T = A_1 \cap A_2 \cap A_3$ . If we call a Set $R$ , then the Array will be: $A = R_1 \cap R_2 \cap \dots \cap R_n$

Table 4-5 Function Description

#### 4.2.2.3.7 Package: alert-distribution-manager

##### Class Diagram : Alert Distribution manager package



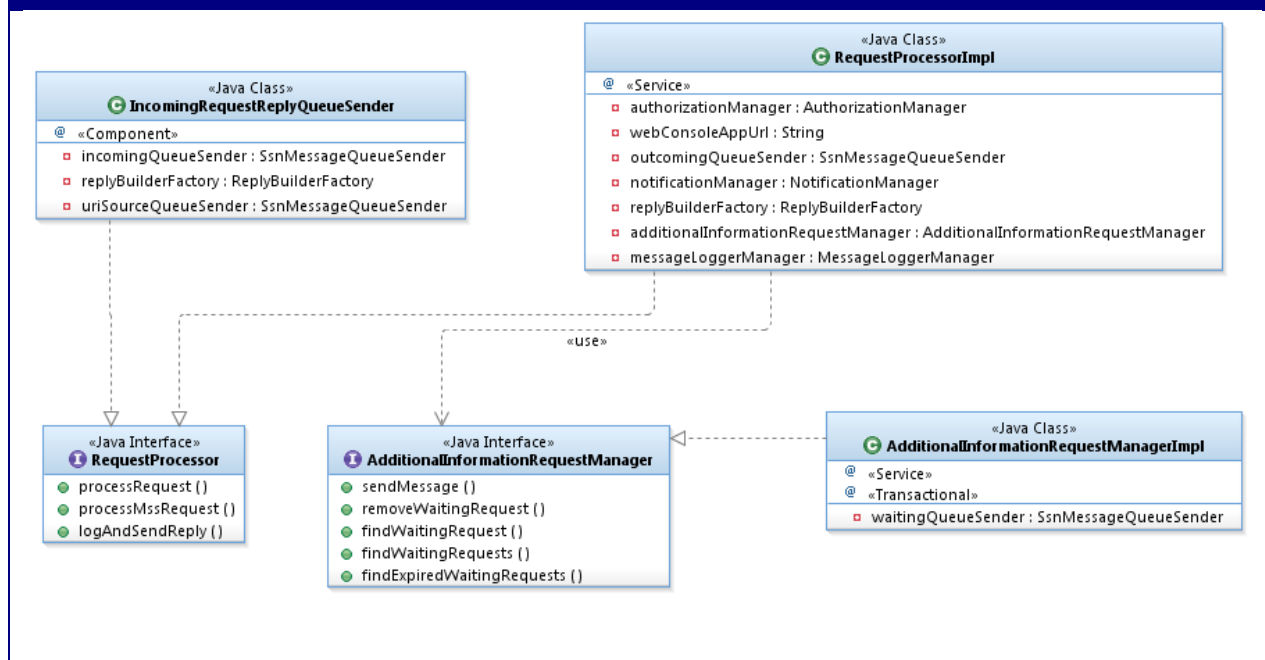
##### Interface

##### AlertDistributionManager

It provides the management of Distributed Alert Notifications and their details.

#### 4.2.2.3.8 Package: request-processor

##### Class Diagram : request-processor



<b>Class</b>	<b>AdditionalInformationRequestManagerImpl</b> Implementation of the AdditionalInfomrationRequestManager interface that provide access to the wait JMS queue using the SsnMessageQueueSender.
<b>Class</b>	<b>RequestProcessorImpl</b> Implementation of the RequestProcessor interface that process an information request.
<b>Interface</b>	<b>RequestProcessor</b> An interface implemented by classes that are capable of processing an information request. There are two implementations of the interface:  IncomingRequestReplyQueueSender: which is used to send a request to the incoming queue.  RequestProcessorImpl: which is used to actually process a request from the incoming queue.
<b>Class</b>	<b>AdditionalInformationRequestManager</b> Implementation of the AdditionalInfomrtaiionRequestProcessor interface that process an information request.
<b>Interface</b>	<b>IncomingRequestReplyQueueSenderProcessor</b> Implementation of the RequestProcessor interface that process an information request.

#### 4.2.2.3.9 Package: expired-request-processor

##### Class Diagram : expired-request-processor

## Class Diagram : expired-request-processor



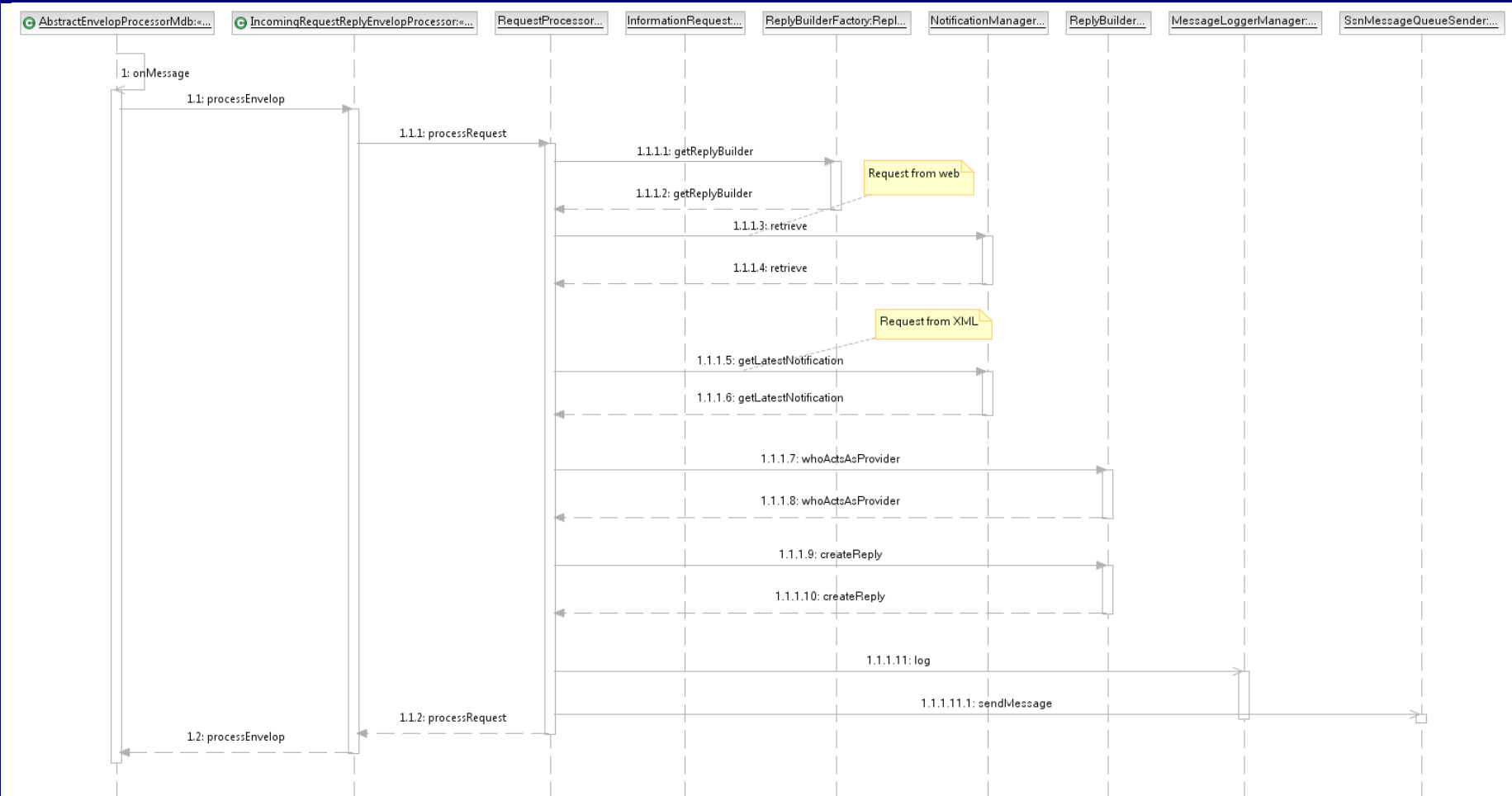
<b>Class</b>	<b>AdditionalInformationRequestManagerImpl</b> Implementation of the AdditionalInfomrationRequestManager interface that provide access to the wait JMS queue using the SsnMessageQueueSender.
<b>Class</b>	<b>ExpiredRequestProcessorImpl</b> Implementation of the ExpiredRequestProcessor

**Class Diagram : expired-request-processor**

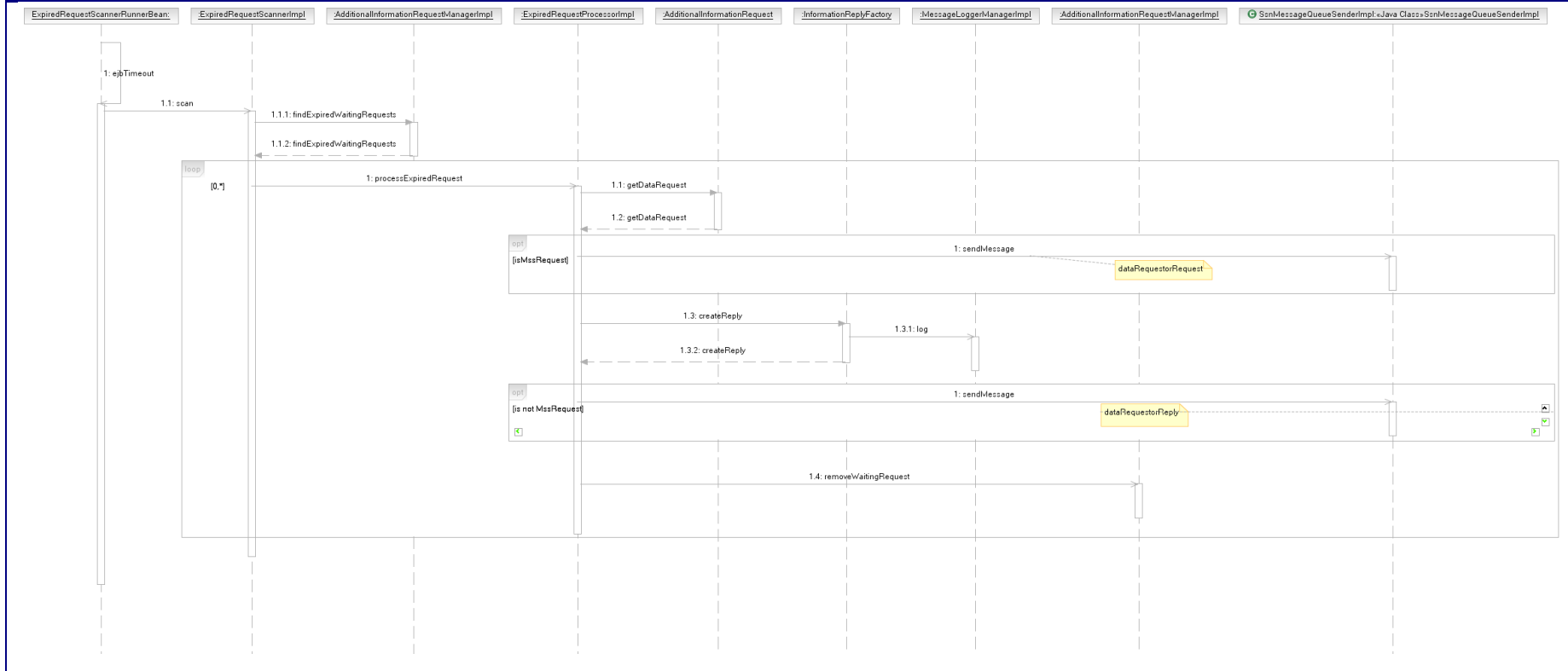
	interface that process an expired request due to time out.
<b>Class</b>	<b>ExpiredRequestScannerImpl</b> Implementation of the ExpiredRequestScanner interface that scans the wait queue for expired requests.
<b>Interface</b>	<b>ExpiredRequestScannerRunner</b> An interface implemented by classes to trigger the scanning of wait queue for expired requests.



### Sequence Diagram : Process Request Reply from EIS



### Sequence Diagram: Process Expired Request from EIS



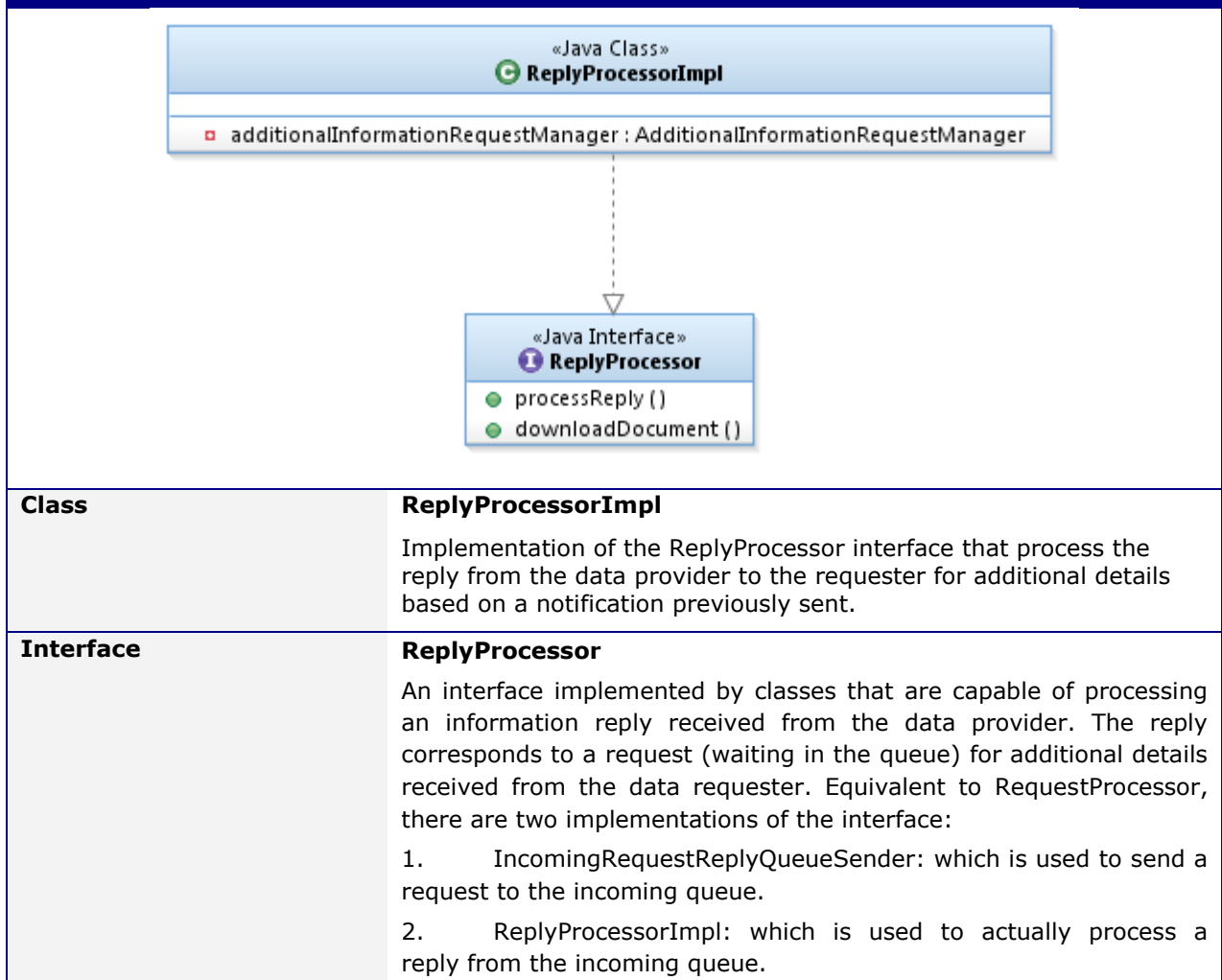
# EUROPEAN COMMISSION

## EUROPEAN MARITIME SAFETY AGENCY

Cais Do Sodré 1249-206 Lisbon, Portugal

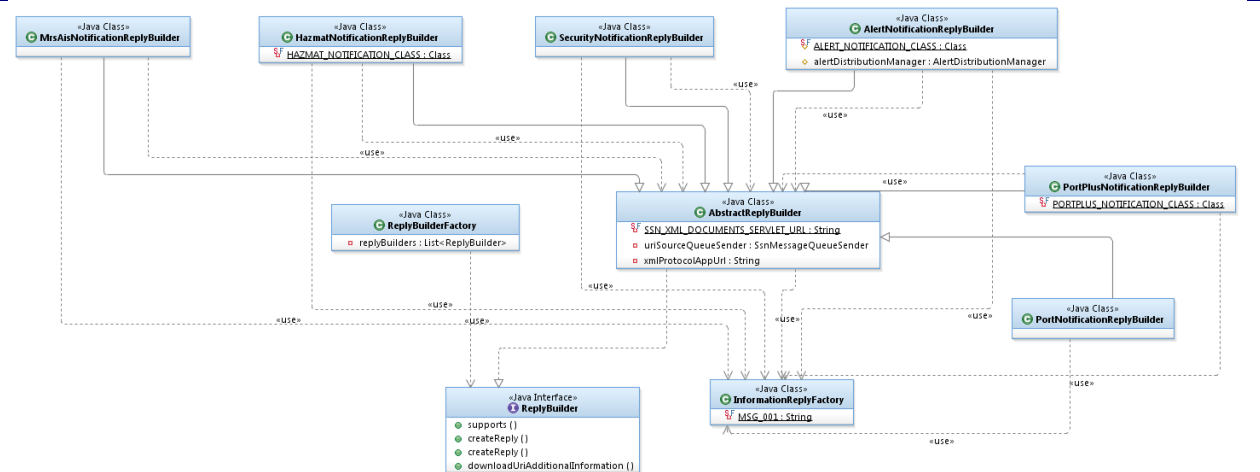
### 4.2.2.3.10 Package: reply-processor

#### Class Diagram : reply-processor



### 4.2.2.3.11 Package: reply-builder

## Class Diagram : reply-builder



<b>Class</b>	<b>AbstractReplyBuilder</b> An abstract implementation of the ReplyBuilder interface that implements the "createReply" method.
<b>Class</b>	<b>ReplyBuilderFactory</b> A factory class used to create the ReplyBuilder.
<b>Class</b>	<b>InformationReplyFactory</b> An abstract factory class used to create the reply upon a request; i.e. a normal and/or an exception (e.g. InformationNotFoundReply) reply.
<b>Interface</b>	<b>ReplyBuilder</b> Indicates whether the current builder can handle the given request. There are two types of replies: the one build based on the reply sent by the data provider and the one build by SSN based on its own resources.
<b>Class</b>	<b>AlertNotificationReplyBuilder</b> An implementation of the ReplyBuilder interface for the Alert notification requests. The reply is build based on the response received by the data provider.
<b>Class</b>	<b>MrsAisNotificationReplyBuilder</b> An implementation of the ReplyBuilder interface for the Ship (MRS/AIS) notification requests. The reply is build based on the response received by the data provider.
<b>Class</b>	<b>HazmatNotificationReplyBuilder</b> An implementation of the ReplyBuilder interface for the Hazmat notification requests. The reply is build based on the response received by the data provider.
<b>Class</b>	<b>PortNotificationReplyBuilder</b> An implementation of the ReplyBuilder interface for the port notification requests. For any port notification SSN can build a reply without interacting with the notification sender.

## Class Diagram : reply-builder

### Class

### PortPlusNotificationReplyBuilder

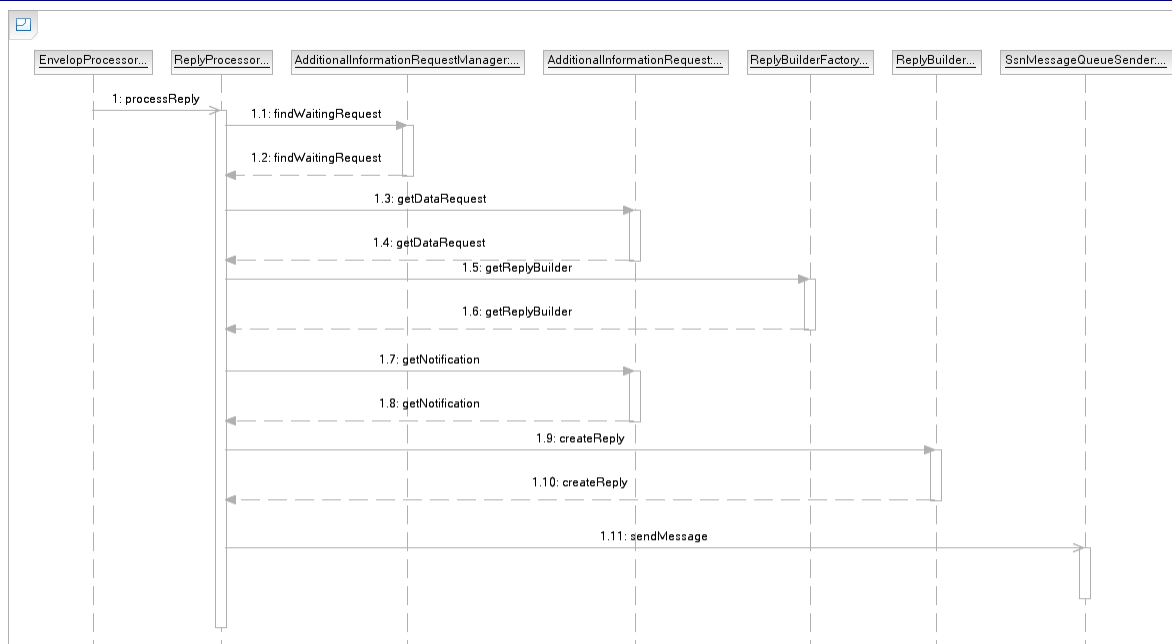
An implementation of the ReplyBuilder interface for the PortPlus notification requests. The reply is build based on the response received by the data provider.

### Class

### SecurityNotificationReplyBuilder

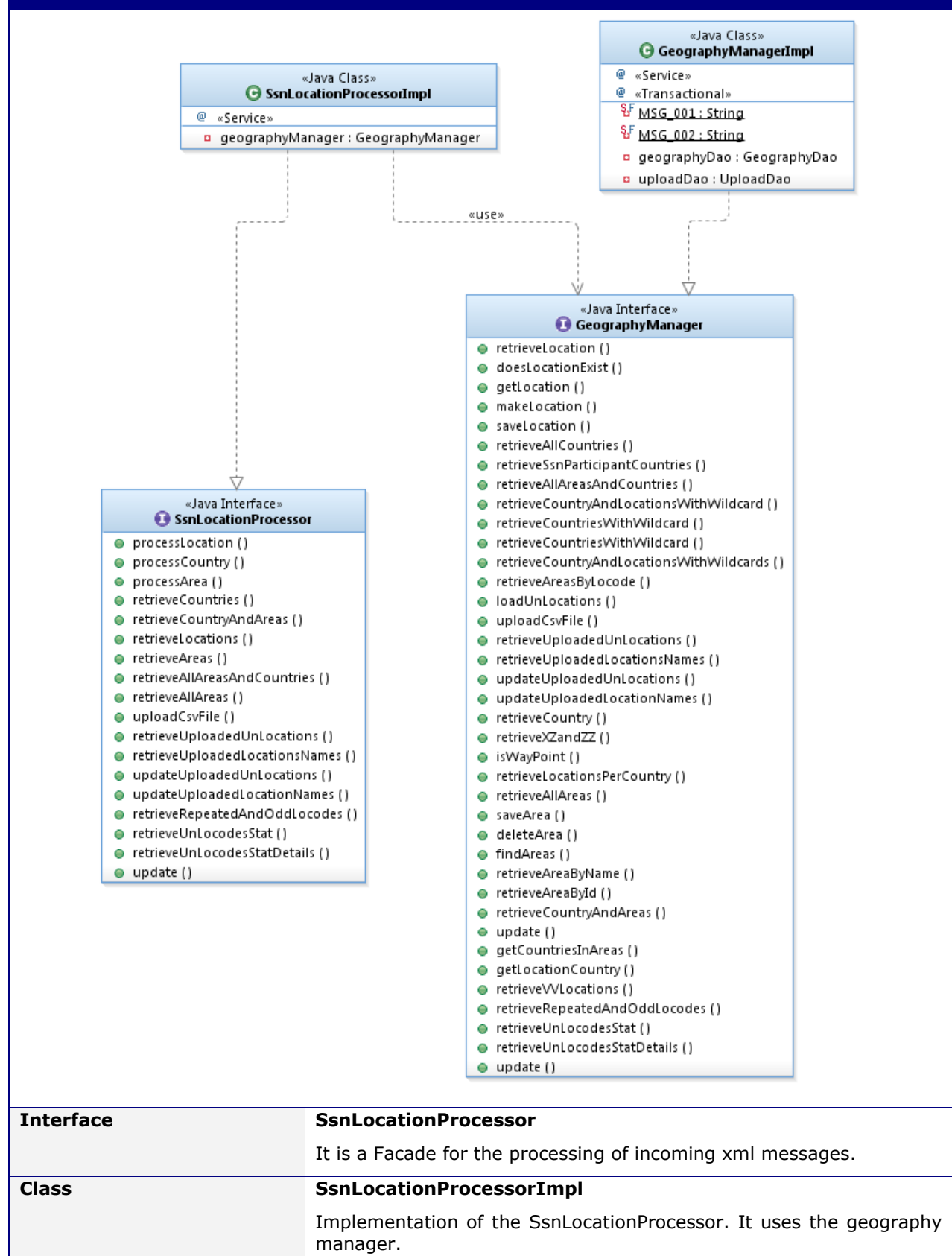
An implementation of the ReplyBuilder interface for the Security notification requests. The reply is build based on the response received by the data provider.

## Sequence Diagram : Reply Processor



#### 4.2.2.3.12 Package: geography-manager

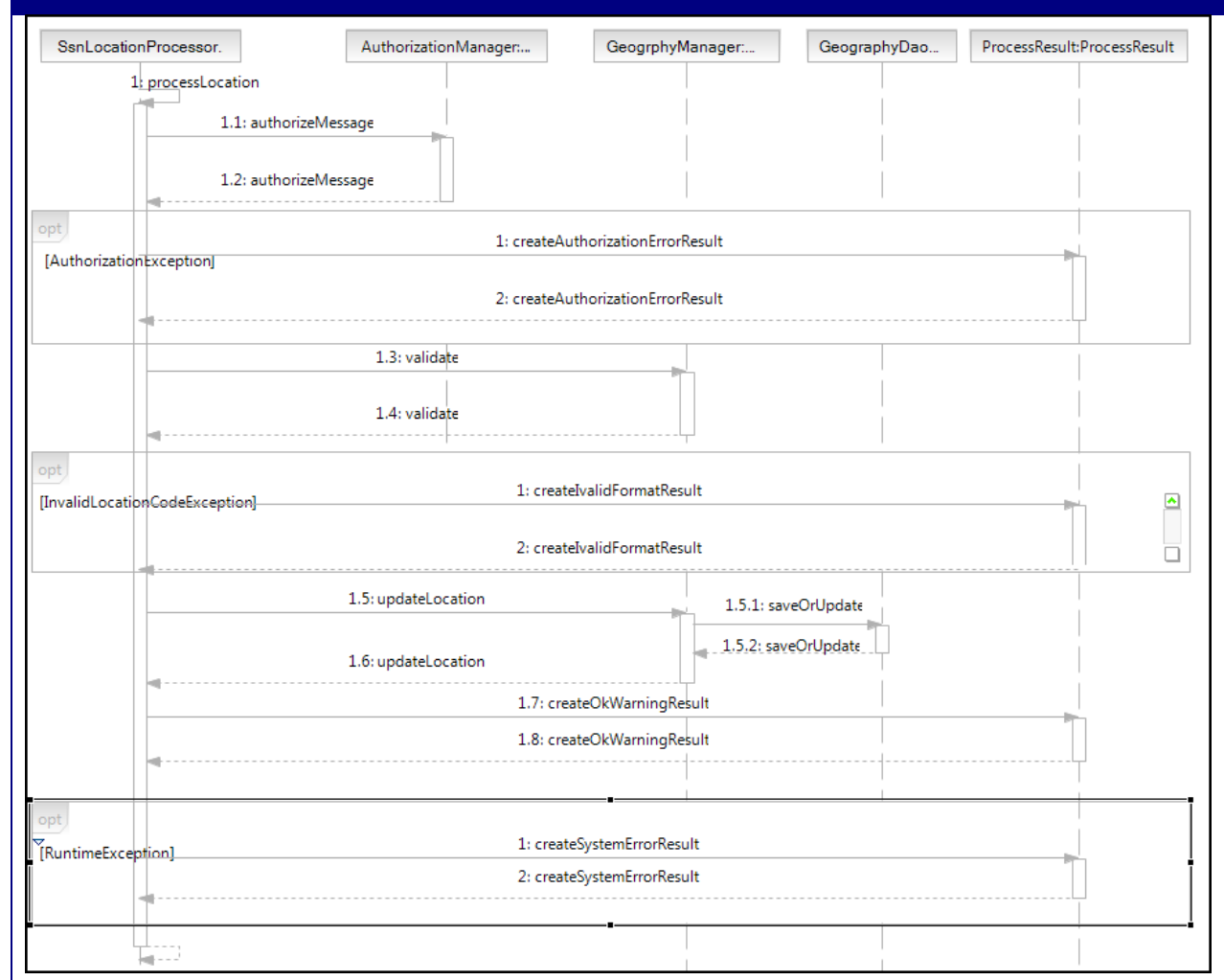
**Class Diagram : geography-manager**



### Class Diagram : geography-manager

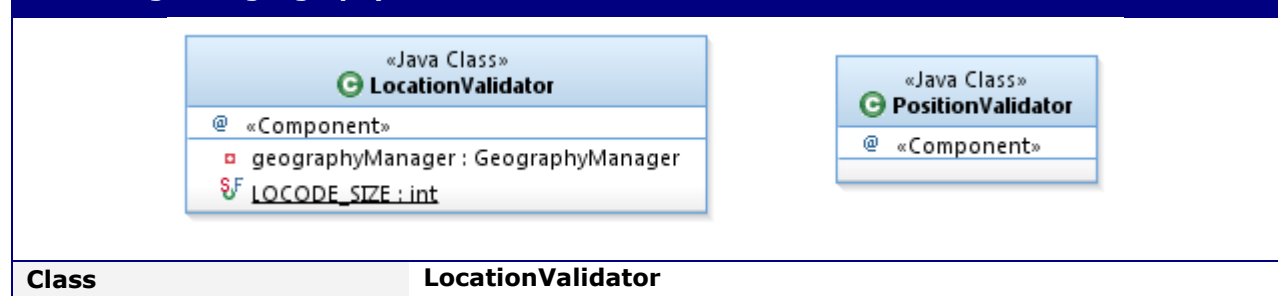
<b>Class</b>	<b>GeographyManagerImpl</b> Default implementation of the LOCODE management processor.
<b>Interface</b>	<b>GeographyManager</b> Defines the processing logic of the LOCODEs.

### Sequence Diagram : Process LocationNotification



#### 4.2.2.3.13 Package: geography-validation

### Class Diagram : geography-validation



**Class Diagram : geography-validation**

	The Location business rules.
<b>Class</b>	<b>PositionValidator</b> The position business rules.

*4.2.2.3.14 Package: organisation-manager*

**Class Diagram : organisation-manager**



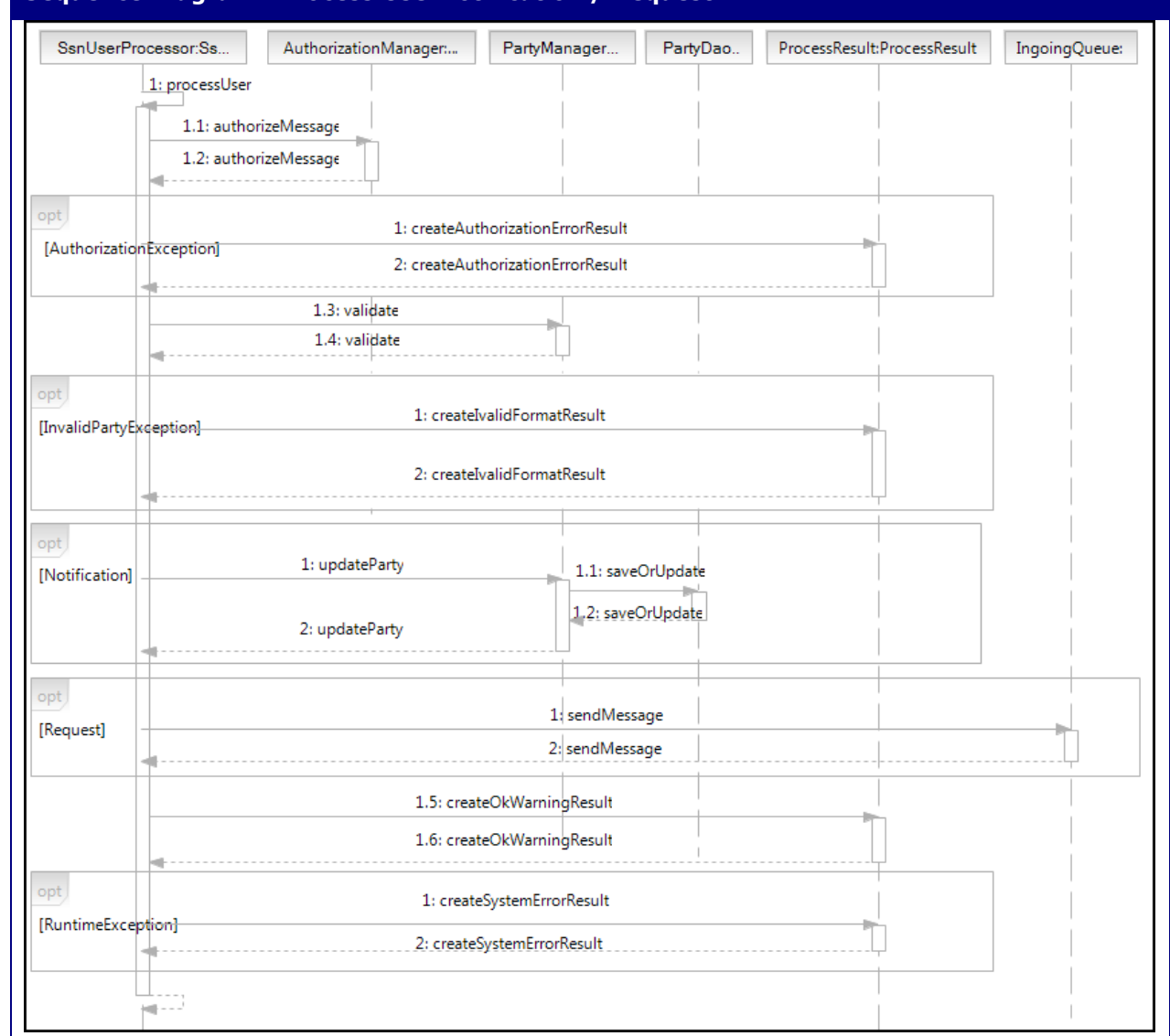
## Class Diagram : organisation-manager



### Class Diagram : organisation-manager

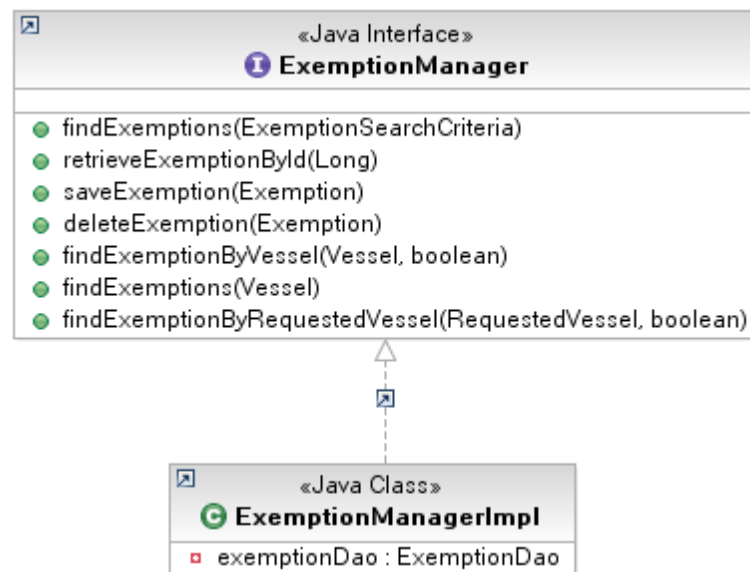
<b>Interface</b>	<b>SsnUserProcessor</b> It is a Facade for the processing of incoming xml messages.
<b>Class</b>	<b>SsnUserProcessorImpl</b> Implementation of the SsnUserProcessor. It uses the party manager.
<b>Interface</b>	<b>PartyManager</b> Defines the processing logic of the parties (SSN user management).
<b>Class</b>	<b>PartyManagerImpl</b> Default implementation of the parties management processor.
<b>Class</b>	<b>PartyNotFoundException</b> Exception is thrown when a requested party is not defined in the database.

### Sequence Diagram : Process UserNotification / Request



#### 4.2.2.3.15 Package: exemption-manager

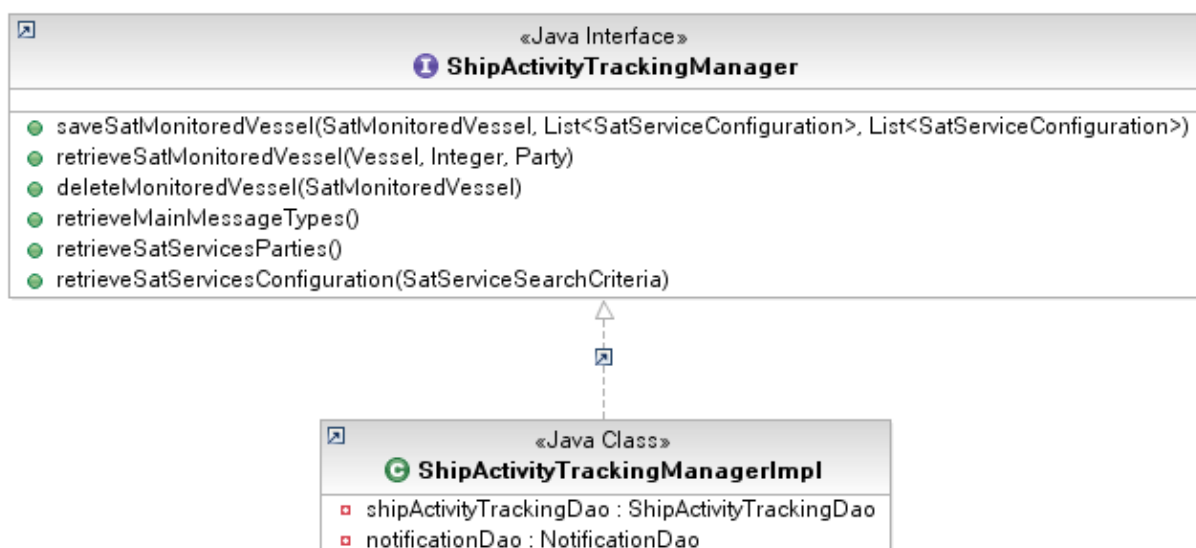
##### Class Diagram : exemption-manager



<b>Interface</b>	<b>ExemptionManager</b> Interface providing the management options for exemptions.
<b>Class</b>	<b>ExemptionManagerImpl</b> Implementation of the ExemptionManager interface.

#### 4.2.2.3.16 Package: sat-manager

##### Class Diagram : sat-manager



<b>Interface</b>	<b>ShipActivityTrackingManager</b> Interface represents the manager for Sat service configurations and reporting.
------------------	--

**Class Diagram : sat-manager**

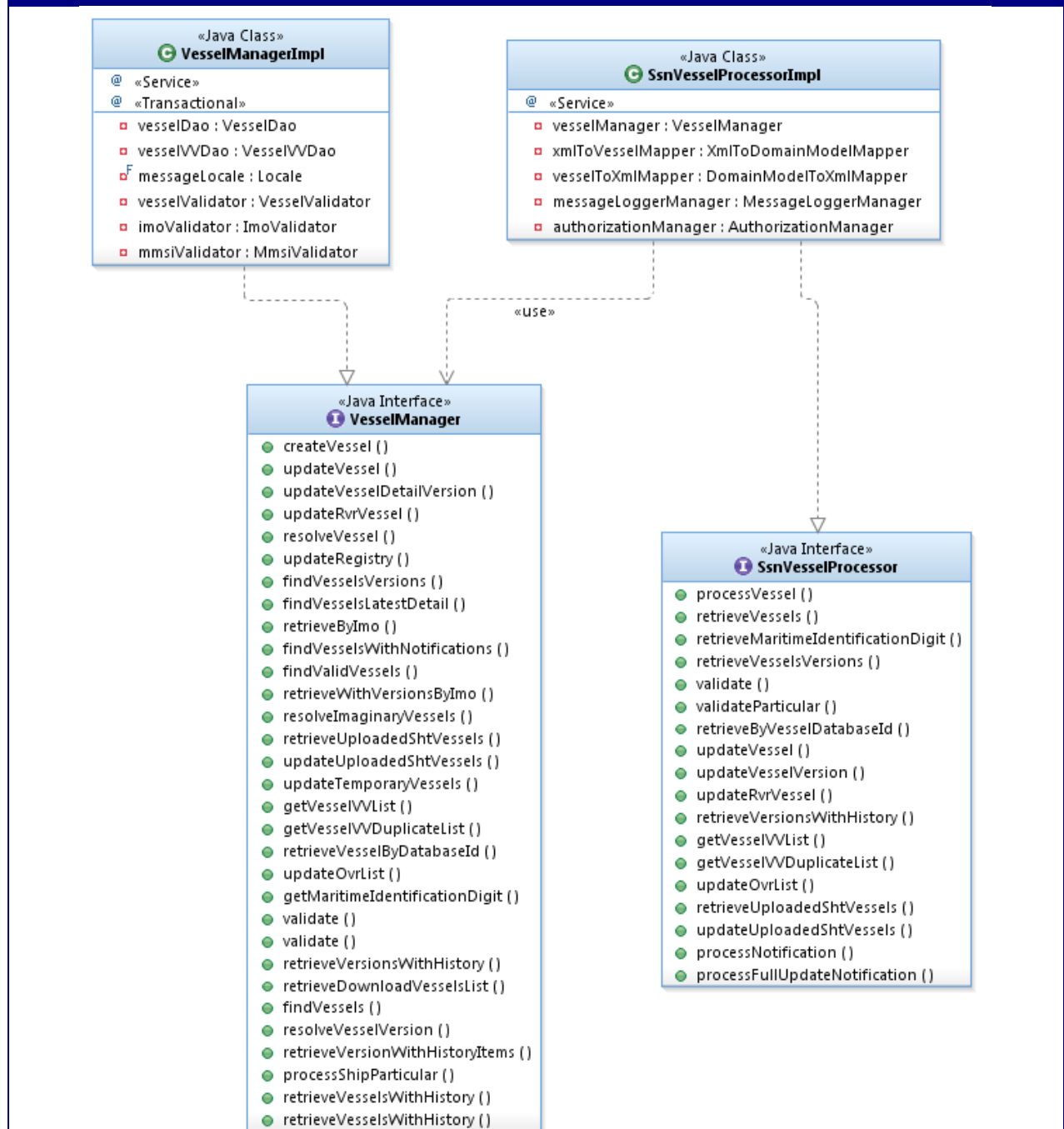
**Class**

**ShipActivityTrackingManagerImpl**

Implementation of the ShipActivityTrackingManager interface.

#### 4.2.2.3.17 Package: vessel-manager

##### Class Diagram : vessel-manager

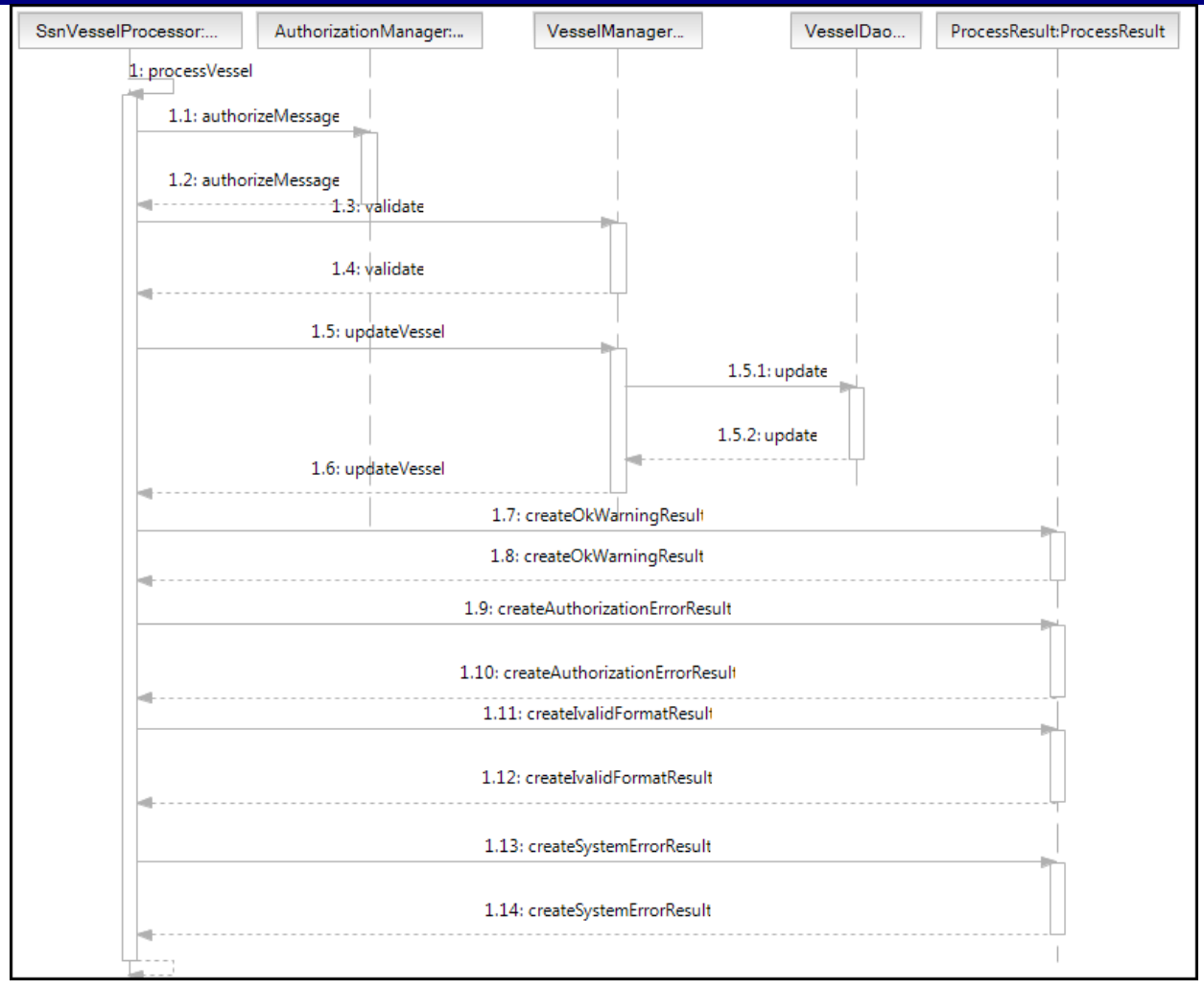


<b>Interface</b>	<b>SsnVesselProcessor</b> It is a Facade for the processing of incoming xml messages. It uses the vessel manager.
<b>Class</b>	<b>SsnVesselProcessorImpl</b> Implementation of the SsnVesselProcessor.
<b>Interface</b>	<b>VesselManager</b> Interface providing the management options for vessel registry.
<b>Class</b>	<b>VesselManagerImpl</b>

## Class Diagram : vessel-manager

Implementation of the VesselManager.

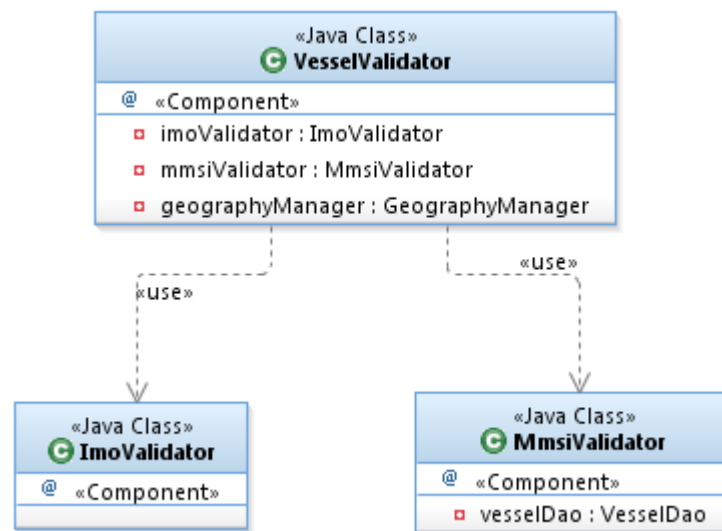
## Sequence Diagram : Process Vessel Notification



### 4.2.2.3.18 Package: vessel-validation

## Class Diagram : vessel-validation

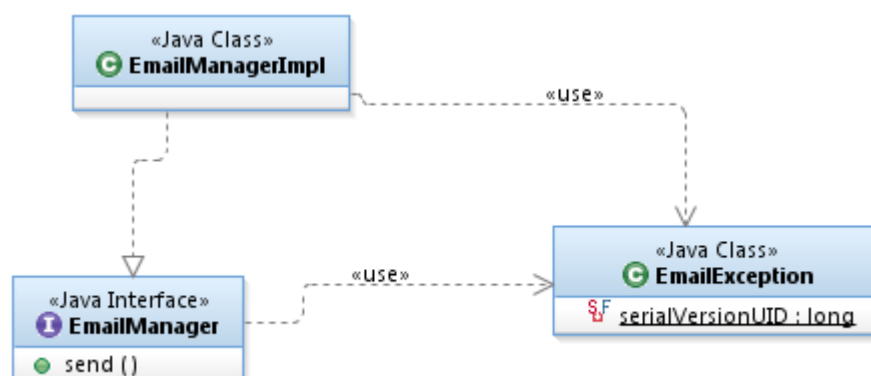
#### Class Diagram : vessel-validation



<b>Class</b>	<b>VesselValidator</b> The vessel business rules.
<b>Class</b>	<b>ImoValidator</b> The vessel IMO number business rules.
<b>Class</b>	<b>MmsiValidator</b> The vessel MMSI number business rules.

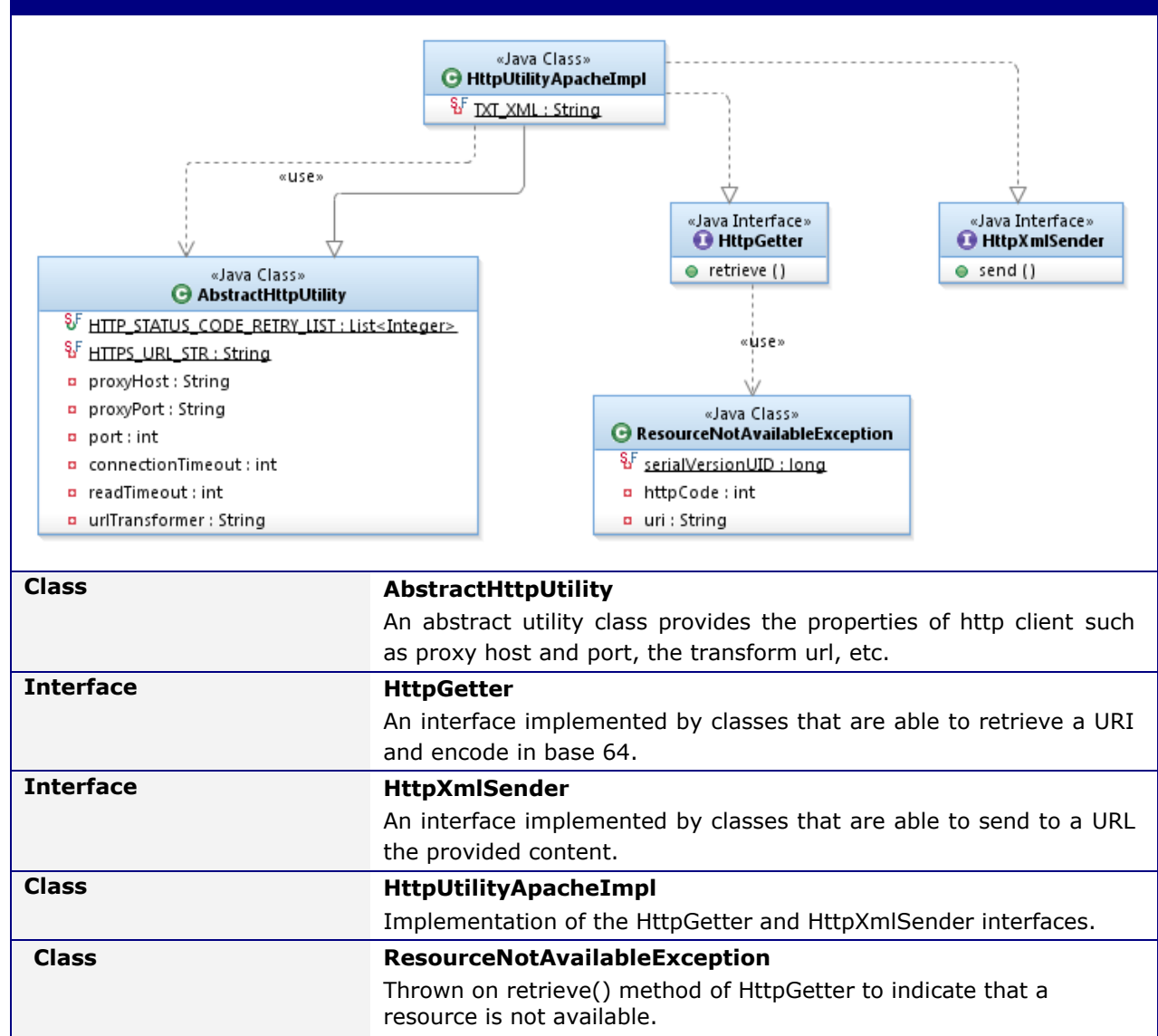
#### 4.2.2.4 Module: ssn-support

#### Class Diagram : support- email



<b>Interface</b>	<b>EmailManager</b> Interface responsible to send emails to SSN users.
<b>Class</b>	<b>EmailManagerImpl</b> Implementation of the EmailManager interface.
<b>Class</b>	<b>EmailException</b> Thrown to indicate failure on email message creation/send.

**Class Diagram : support- http**



**Class Diagram : wsclient**

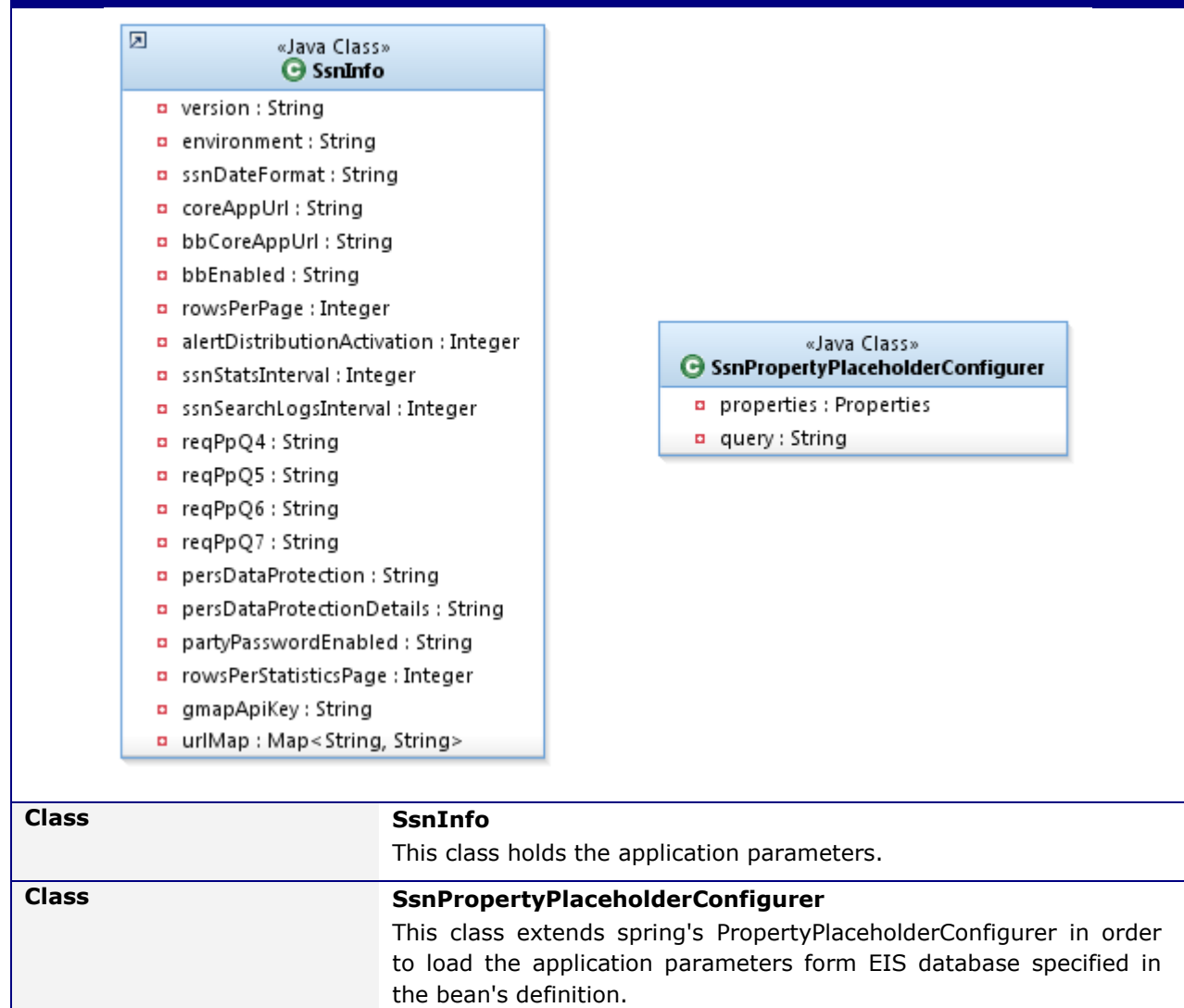




### Class Diagram : wsclient

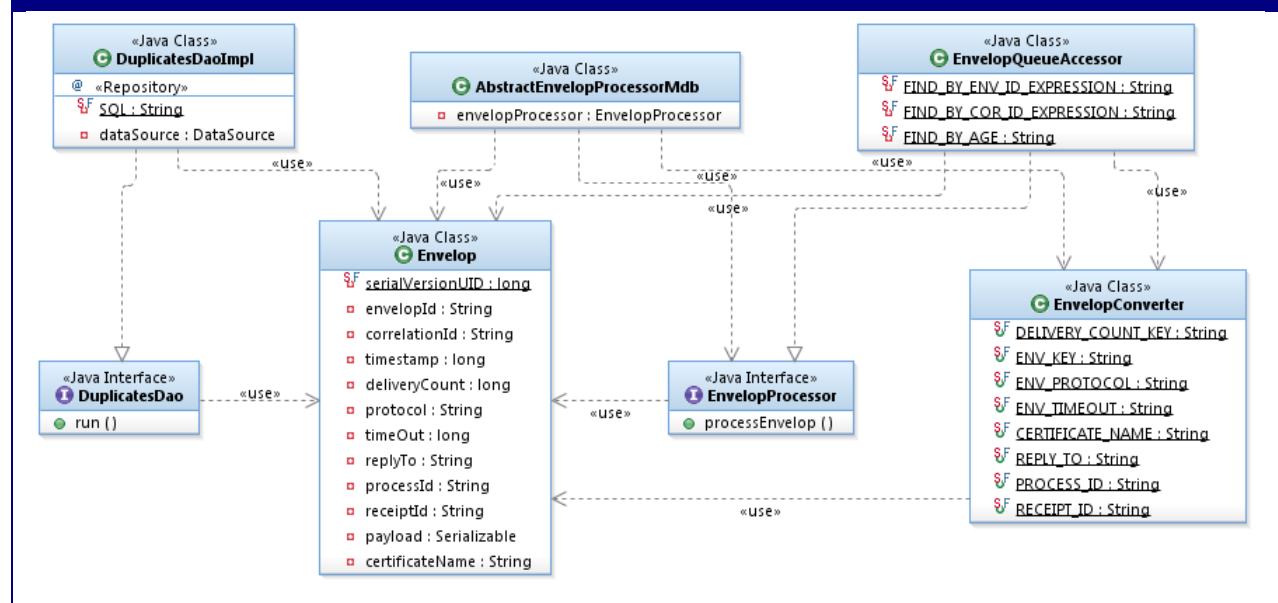
via HTTP to data requestors.

### Class Diagram : support- info



### Class Diagram : support-jms

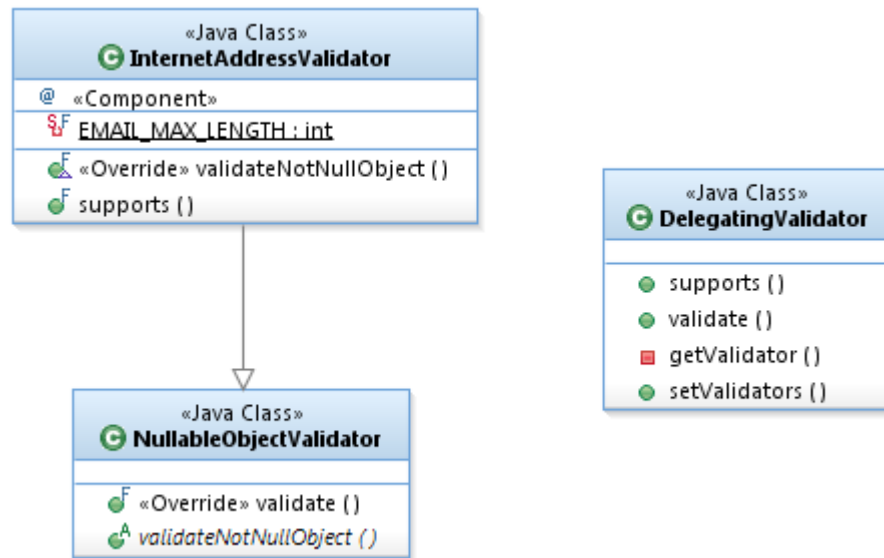
### Class Diagram : support-jms



<b>Class</b>	<b>Envelop</b> This class it is used to model the information carried in a JMS message.
<b>Class</b>	<b>EnvelopConverter</b> This class implements the message converter; it creates an envelop from a JMS message and vice versa.
<b>Class</b>	<b>EnvelopProcessorRetryCountAdvice</b> This class implements the interceptor handles the failures of JMS message processing. In case of IOException on message processing of the SSN outgoing JMS queue items, the message remains on the queue to be reprocessed – the “retry count” application parameter is taken also into account.
<b>Class</b>	<b>AbstractEnvelopProcessorMdb</b> An abstract processor implements the JMS Message Listener interface (onMessage() method).
<b>Interface</b>	<b>EnvelopProcessor</b> An interface defines the JMS queues’ message processing.
<b>Class</b>	<b>EnvelopQueueAccessor</b> Implements the EnvelopProcessor interface; it is used by SsnMessageQueueSenderImpl.

### Class Diagram : support- validation

#### Class Diagram : support- validation



<b>Class</b>	<b>NullableObjectValidator</b> An abstract implementation of the validator interfaces that checks if the given object is null. If it is, it returns without reporting any errors. If the given object is not null it calls the validateNotNullObject(Object, Errors) to perform the actual validation. The responsibility to check if the given object is null lies to the caller of this class.
<b>Class</b>	<b>InternatAddressValidator</b> An abstract implementation of the validator interfaces that checks the email.
<b>Class</b>	<b>DelegatingValidator</b> An abstract implementation of the validator interfaces that resolves the SSN validator according to the object to be validated.

### 4.2.3 Security on the ssn-core-app

The ssn-core-app application security relies on:

- The system level security provided by the runtime environment (SSL and Client Certificate).
- The application level security offered by ssn-core-app.

More specifically on the application level security, each message carries the credentials of its sender. So ssn-core-app uses only this form of credentials to authenticate a message.

The **ssn-core-app** application uses the sender of the message and the type of the message to perform authorization decisions.

The authentication and authorization methods are implemented by the **AuthorizationManagerImpl** class – located under the package **ssn.message.manager**.

Limitations: **ssn-core-app** must be compliant with the current version of the SSN protocol for exchanging XML messages. So, **ssn-core-app**:

- Must use a rather weak form of credentials (only username) in order to authenticate messages

Cannot leverage advanced non-repudiation techniques such as XML Signature<sup>1</sup> and XML Encryption<sup>2</sup>.

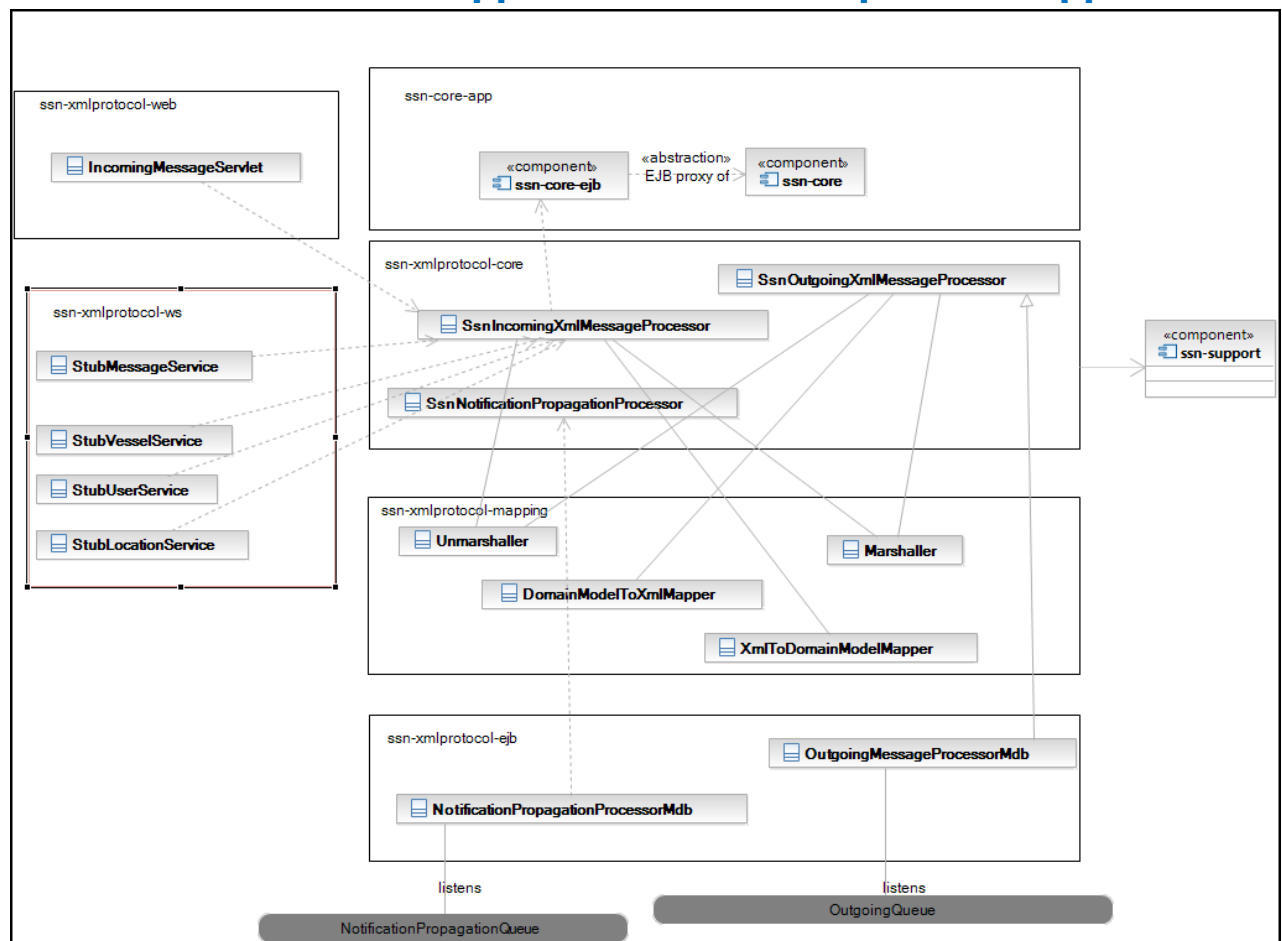
### 4.3 SSN Console Application - ssn-console-app

The application is constituted from the following web applications:

1. **ssn-admin-console**: It is a Web application, which will be used by the EIS users for the management of EIS assets.
2. **send-notification-console**: This web application is used for sending Notifications.
3. **find-notification-console**: This web application is used for sending Requests for details.
4. **reports-statistics-console**: This web application is used for EIS reports generation.
5. **information-download-console**: This web application is used by the SSN users to download notification forms.

The web applications do not contain distinguishable components and is simply a UI. It is simply using the functionality provided by the ssn-core-app.

### 4.4 SSN XML Protocol Application - ssn-xmlprotocol-app



<sup>1</sup>XML Signatures provide integrity, message authentication, and/or signer authentication services for data of any type, whether located within the XML that includes the signature or elsewhere. (ref: W3C Recommendation 12 February 2002).

<sup>2</sup>Requirements on the encryption syntax, data model, format, cryptographic processing, and external requirements and coordination. (ref: W3C Note 04 March 2002)

**Figure 4-5 SSN XML Protocol Application - ssn-xmlprotocol-app**

The followings modules constitute the application:

1. **ssn-xmlprotocol-mapping**: It provides the functionality of representing the XML messages in Java Objects - Data Transfer Objects (DTO's) - and the mapping of these objects to domain model - Domain Objects (DO's)- of ssn-core and vice versa.  
  
DTO's are also an important part of the design in an SOA environment where the Domain object model structurally is not compatible with the messages that are received and sent from a business service. The messages are typically defined and maintained in as XML Schema Definition documents (XSD's) and it's a common practice to write (or code generate) DTO objects from the XSD's and use them for data (message) transfer purposes between domain and SOA service layers.
2. **ssn-xmlprotocol-core**: It provides the management of the incoming and outgoing Xml messages independent from the channel of communication. It uses the ssn-xmlprotocol-mapping for the mapping of the XML messages to domain model object and it calls directly ssn-core-app in order to process the messages.
3. **ssn-xmlprotocol-web**: It receives incoming XML messages from the HTTP communication channel and delivers them to ssn-xmlprotocol-core.
4. **ssn-xmlprotocol-ws**: It receives incoming SOAP messages from the HTTP communication channel and delivers them to ssn-xmlprotocol-core.
5. **ssn-xmlprotcol-ejb**: It receives - asynchronously -
  - a. outgoing messages from ssn-core-app and dispatches them via HTTP to the recipients (data Requesters),
  - b. propagation messages from ssn-core-app and dispatches them via the mail session (SMTP) to the recipients.

To support the transition from the the XMLRG V2 to the latest V3, SSN will entail 2 distinct XML protocol applications:

- The ssn-xml-protocol that will support the XMLRG V3 and
- The ssn-xml-protocol-v2 that will support the XMLRG V2.

The ssn-xml-protocol-v2 will transparently handle v2 compliant communication with ssn-core-app or any other SSN-EIS components. The rationale behind this decision is that this component will be easily discarded with minimum side-effects and administration cost after the transition period.

V2 messages subject to transition period will be handled by the ssn-xml-protocol-v2 are:

V2 MS2SSN\_Ship\_Not  
V2 MS2SSN\_PortPlus\_Not  
V2 MS2SSN\_ShipCall\_Req  
V2 SSN2MS\_ShipCall\_Req  
V2 MS2SSN\_ShipCall\_Res  
V2 SSN2MS\_ShipCall\_Res

In addition, messages common to XMLRG V2 and V3 will also be handled by the V2 ssn-xml-protocol-v2. I.e.:

MS2SSN\_Alert\_Not  
MS2SSN\_IncidentDetail\_Not  
MS2SSN\_Alert\_Req.xml  
MS2SSN\_Alert\_Res.xml  
MS2SSN\_IncidentReport\_Req.xml  
SSN2MS\_IncidentDetail\_Tx  
SSN2MS\_IncidentDetail\_Tx\_Ack  
SSN2MS\_Alert\_Req.xml

SSN2MS\_Alert\_Res.xml  
SSN2MS\_IncidentReport\_Res.xml

## 4.4.1 SSN XML Protocol Main Components

### 4.4.1.1 SSN XML Protocol Mapping Module - *ssn-xmlprotocol-mapping*

The basic components are listed in Table 4-6.

	Component	Description
1	Marshaller	It provides the possibility of representing a Java object in XML.
2	Unmarshaller	It provides the possibility of representing XML in a Java object.
3	DomainModelToXmlMapper	It provides the correspondence of a domain model object in ssn-core to a Java object representing XML.
4	XmlToDomainModelMapper	It provides the correspondence of a Java object that represents XML in the domain model of ssn-core.

**Table 4-6 *ssn-xmlprotocol-mapping***

### 4.4.1.2 SSN XML Protocol Core module: *ssn-xmlprotocol-core*

The basic components are listed in Table 4-7.

	Component	Description
1	SsnIncomingXmlMessageProcessor	It provides the synchronous management of incoming XML messages - independent from the channel of communication they were received.
2	SsnOutgoingXmlMessageProcessor	Transforms outgoing messages in XML independent from the channel of communication in which they will be delivered.
3	SsnNotificationPropagationProcessor	Transforms notification messages to e-mail messages to be propagated via SMTP email server.

**Table 4-7 *ssn-xmlprotocol-core***

### 4.4.1.3 SSN XML Protocol Web module - *ssn-xmlprotocol-web*

The basic component is listed in Table 4-8.

	Component	Description
1	IncomingMessageServlet	It receives incoming XML messages from the HTTP communication channel and delivers them to <i>ssn-xmlprotocol-core</i> .

**Table 4-8 *ssn-xmlprotocol-web***

### 4.4.1.4 SSN XML Protocol EJB module - *ssn-xmlprotocol-ejb*

The basic components are listed in Table 4-9.

	Component	Description
1	OutgoingMessageProcessorMdb	– Message Driven Bean that processes the messages from the Outgoing Queue. It uses <i>ssn-xmlprotocol-core</i> in order to convert them to XML and dispatch them to the recipients using HTTP.

	Component	Description
2	NotificationPropagationProcessorMdb	<ul style="list-style-type: none"><li>– Message Driven Bean that processes the messages from the Notification PropagationQueue. It uses ssN-xmlprotocol-core in order to convert them to e-mail messages and send them to the recipients using JNDI mail session (SMTP).</li></ul>

**Table 4-9 ssN-xmlprotocol-ejb**

#### **4.4.1.5 SSN XML Protocol WebServices module - ssN-xmlprotocol-ws**

The ssN-xmlprotocol-ws module will be included into the existing ssN-xmlprotocol-app application because there are common parts like ssN-xmlprotocol-core or ssN-xmlprotocol-mapping. Furthermore, the ssN-xmlprotocol-ws will be implemented using Spring-WS which is integrated with Spring in accordance to the existing application modules.

The SOAP message is dispatched to an Endpoint implementation through mappings (EndpointMappings) that bound incoming SOAP messages to custom Endpoints ( depending on the SOAP message itself ). After retrieving the Endpoint, Spring-WS is searching for Adapters (EndpointAdapter) supporting the acquired endpoint. EndpointAdapter defines the way a SOAP message is converted in order to be handled by the endpoint.

The endpoint uses the SsnIncomingXmlMessageProcessor for the management of the incoming messages. The ssN-xmlprotocol-mapping is used in order to map JAXB elements to SSN Domain objects.

#### **4.4.2 UML Class and Sequence Diagrams**

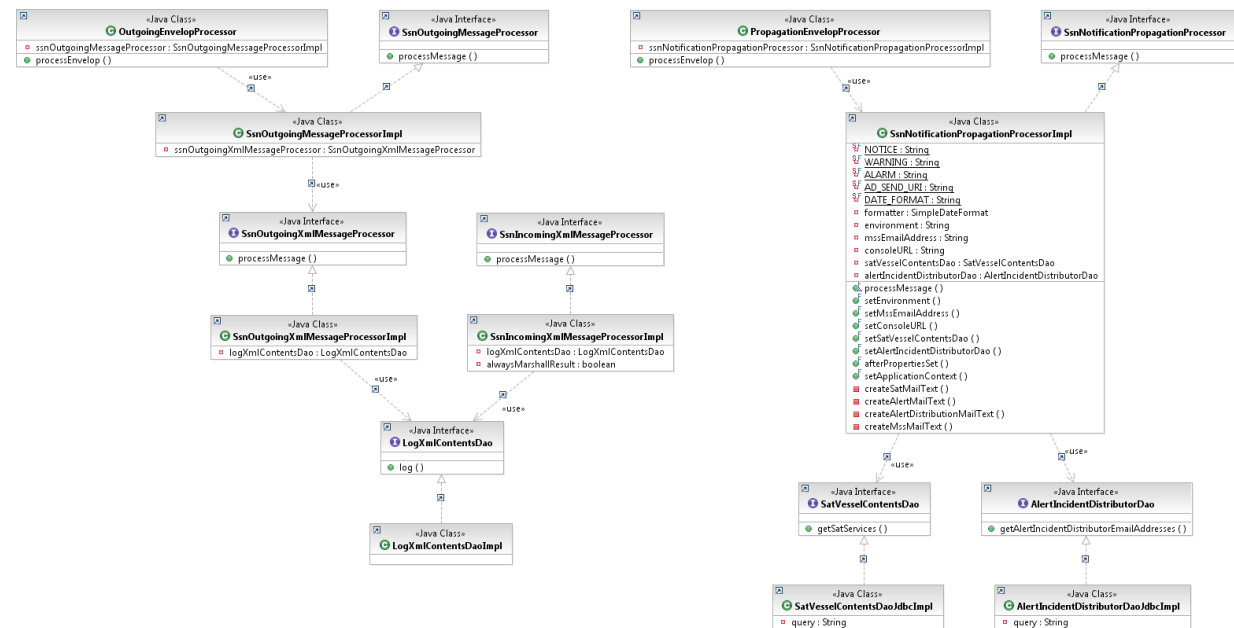
This section covers the architectural significant elements of the design model. It presents the definition of the most significant classes that will implement the requested functionality, organised into packages.

The UML Sequence Diagrams also presented in this section, associate classes and depict the overall flow of control within the system components.

The classes are organised in packages according to the functionality they provide. A package is a general-purpose model element that organizes model elements into groups. Each package contains a set of classes and interfaces, representing what will become components in the implementation.

#### 4.4.2.1 Package: xmlprotocol core

##### Class Diagram : xmlprotocol core



<b>Class</b>	<b>OutgoingEnvelopProcessor</b> It implements the EnvelopProcessor to process the messages located at the SSN outgoing queue. It uses the SsnOutgoingMessageProcessor for the message processing.
<b>Interface</b>	<b>SsnOutgoingMessageProcessor</b> Defines the processing of messages located at the SSN outgoing queue.
<b>Class</b>	<b>SsnOutgoingMessageProcessorImpl</b> Default implementation of the SsnOutgoingMessageProcessor interface. It uses the SsnOutgoingXmlMessageProcessor to transform the outgoing message to xml. Then, it sends the xml message using the HttpXmlSender / SoapSender according to the data requestor type.
<b>Interface</b>	<b>SsnOutgoingXmlMessageProcessor</b> Defines an interface that transforms the outgoing message to xml.
<b>Class</b>	<b>SsnOutgoingXmlMessageProcessorImpl</b> Default implementation of the SsnOutgoingXmlMessageProcessor interface. It uses the LogXmlContentsDao to log the xml message to EIS database.
<b>Interface</b>	<b>SsnIncomingXmlMessageProcessor</b> Defines the processing of incoming xml messages.
<b>Class</b>	<b>SsnIncomingXmlMessageProcessorImpl</b> Default implementation of the SsnIncomingXmlMessage Processor interface. This implementation follows the next processing



Class Diagram : xmlprotocol core	
	<p>procedure (see also Sequence Diagram : EIS Incoming XML Message Processor):</p> <ol style="list-style-type: none"> <li>1. Unmarshalls the incoming XML stream into a java object that represents this XML.</li> <li>2. The provided message mapper is being used to map the previously generated java object into a domain model message.</li> <li>3. The provided message processor is being called to process the domain model message and returns a process result.</li> <li>4. The provided process result mapper maps the process result into a java object that represents XML.</li> <li>5. Finally, the provided marshaller marshalls the previously generated object to the XML output stream.</li> </ol> <p>It uses the LogXmlContentsDao to log the incoming xml message and the process result in xml format (SSN_Receipt) to EIS database.</p>
<b>Interface</b>	<p><b>LogXmlContentsDao</b></p> <p>Interface providing methods for logging the xml messages on EIS database</p>
<b>Class</b>	<p><b>LogXmlContentsDaoImpl</b></p> <p>JDBC based implementation of the LogXmlContentsDao.</p>
<b>Class</b>	<p><b>PropagationEnvelopProcessor</b></p> <p>It implements the EnvelopProcessor to process the messages located at the SSN notification propagation queue. It uses the SsnNotificationPropagationProcessor for the message processing.</p>
<b>Interface</b>	<p><b>SsnNotificationPropagationProcessor</b></p> <p>Defines the processing of messages located at the SSN notification propagation queue.</p>
<b>Class</b>	<p><b>SsnNotificationPropagationProcessorImpl</b></p> <p>Default implementation of the SsnNotificationPropagationProcessor interface.</p> <p>It creates an email according to the message</p> <ul style="list-style-type: none"> <li>- Vessel notification with sat service: it uses the SatVesselContentsDao to retrieve the sat services in order to create the corresponding emails</li> <li>- Alert notification: it uses the AlertIncidentDistributorDao to retrieve the email addresses of incident distributors in order to create the corresponding email.</li> <li>- Alert Distribution notification: it creates an email to alert the notification recipients.</li> </ul> <p>It sends the email message using the EmailManager.</p>
<b>Interface</b>	<p><b>SatVesselContentsDao</b></p> <p>A dao that retrieves the sat services.</p>
<b>Class</b>	<p><b>SatVesselContentsDaoJdbcImpl</b></p> <p>JDBC based implementation of the SatVesselsContentsDao interface.</p>

### Class Diagram : xmlprotocol core

#### Interface

#### AlertIncidentDistributorDao

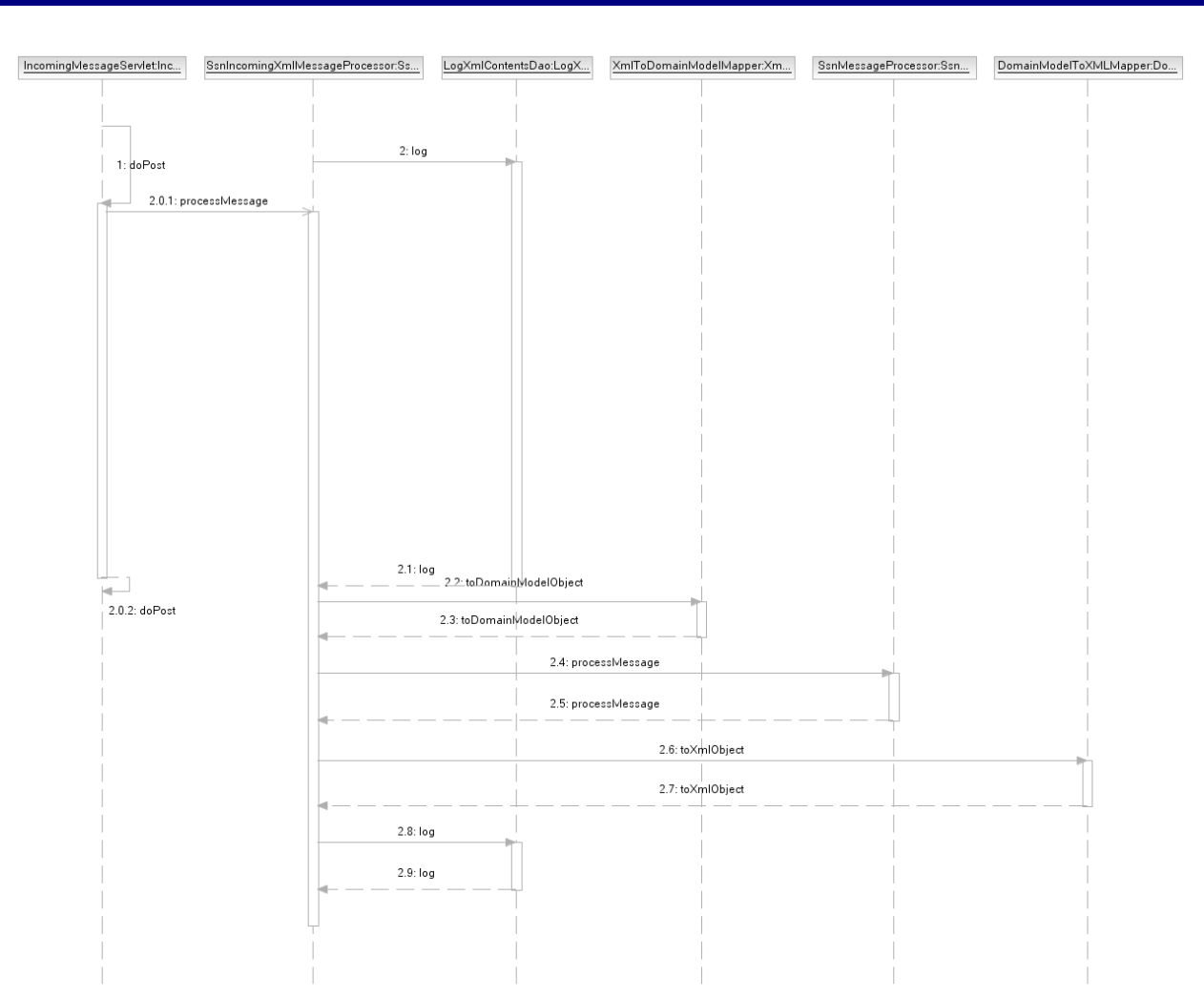
A dao that retrieves the parties with IncidentRepDistributor permission belongs to the country of notification sender.

#### Class

#### AlertIncidentDistributorDaoJdbcImpl

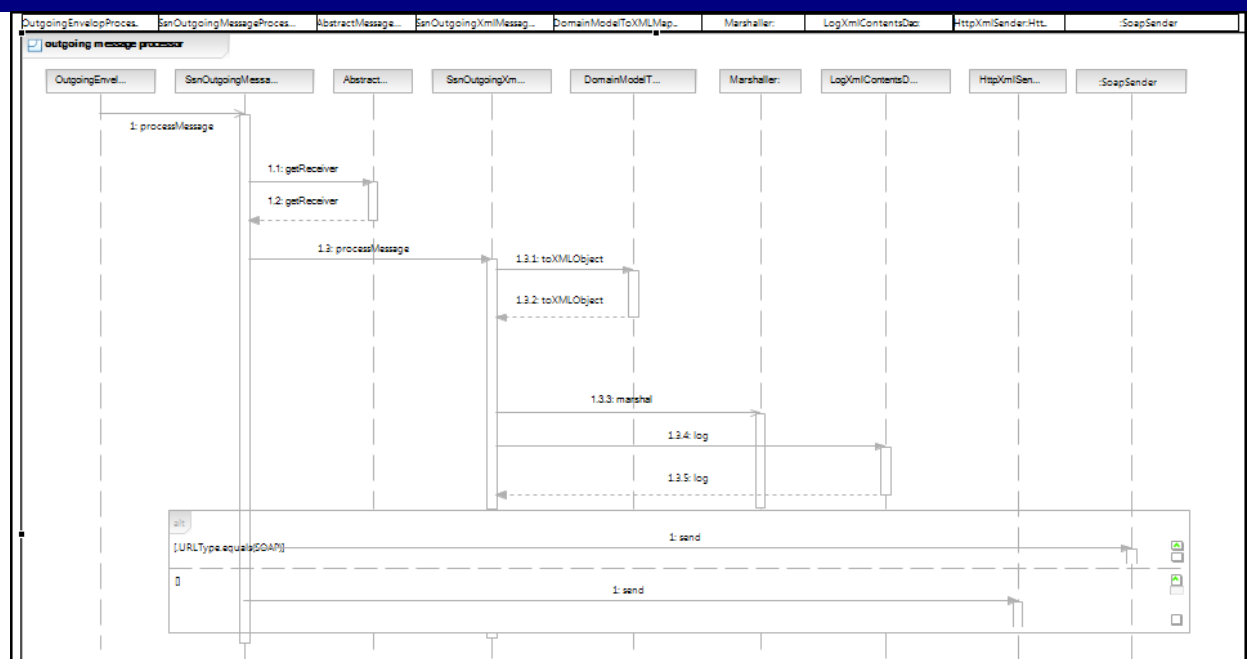
JDBC based implementation of the AlertIncidentDistributorDao interface.

### Sequence Diagram : EIS Incoming XML Message Processor



### Sequence Diagram : EIS Outgoing XML Message Processor

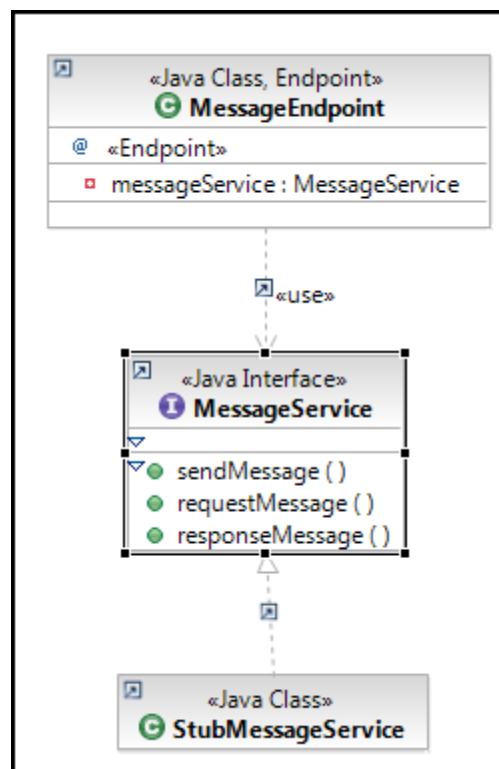
### Sequence Diagram : EIS Outgoing XML Message Processor



#### 4.4.2.2 Package xmlprotocol-ws

##### 4.4.2.2.1 WebService: MessageService

### Class Diagram : MessageService



## Class Diagram : MessageService

<b>Class</b>	<b>MessageEndpoint</b> The payload endpoint handles the incoming SOAP messages. It uses the MessageService to operate. It returns the ProcessResult (Ssn_Receipt) message as response.
<b>Interface</b>	<b>MessageService</b> Provides the business service interface for the incoming messages processing.
<b>Class</b>	<b>StubMessageService</b> The stub implementation of MessageService; it actually delegates the processing to the xml-protocol module processors

## Service Contract : message.wsdl

<b>WSDL</b>	<b>Message.wsdl</b> Example of service contract WSDL file; it is based on ssn.xsd schema (data contract).
<b>Data contract</b>	<b>ssn.xsd</b>

## Notification Messages

The Figure 4-6 shows the messages (SOAP over HTTP) exchanges between the Data Requestors and EIS Service Provider (Web Services) related to the Message Domain as well as the relationship of these messages with the domain classes. More specifically,

- The **MessageNotification (MS2SSN\_<SSN\_type>\_Not)** message is sent by a Member State or system linked to SafeSeaNet (acting as IR Data Provider) in order to **notify** SafeSeaNet that a Member State owns some kind of information about an incident. Message Information is represented by the classes consist the notification package of the domain model.

The IR Data Provider should provide the **acknowledgeCallback** operation where SSN shall send the **ssn\_ir\_notification\_tx\_ack (SSN2MS\_IncidentDetail\_Tx\_Ack)** asynchronously; this message is generated by SSN system and delivered by SsnOutgoingMessageProcessor of the xmlprotocol-core module acting as webservice client.

The role of the IR Recipient is also related with the new IR notifications;

the IR Recipient should provide the **notifyIR** operation where SSN shall send the **ssn\_ir\_notification\_tx (SSN2MS\_IncidentDetail\_Tx)** asynchronously; this message is generated by SSN system and delivered by SsnOutgoingMessageProcessor of the xmlprotocol-core module acting as webservice client. This operation requires a synchronous response **ms\_ir\_notification\_receipt (SSN\_Receipt)**.

- The operations request/response (messages of type **MS2SSN\_<SSN\_type>\_Req, MS2SSN\_<SSN\_type>\_Res**) are not affected.

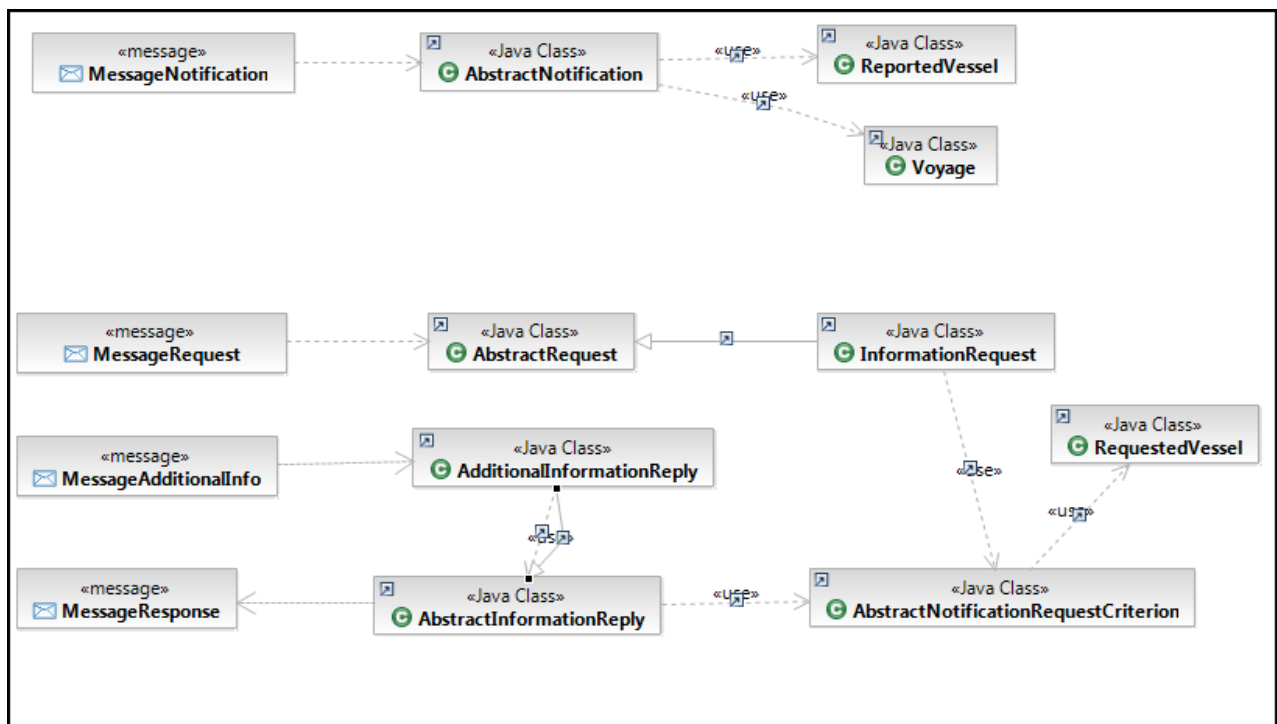
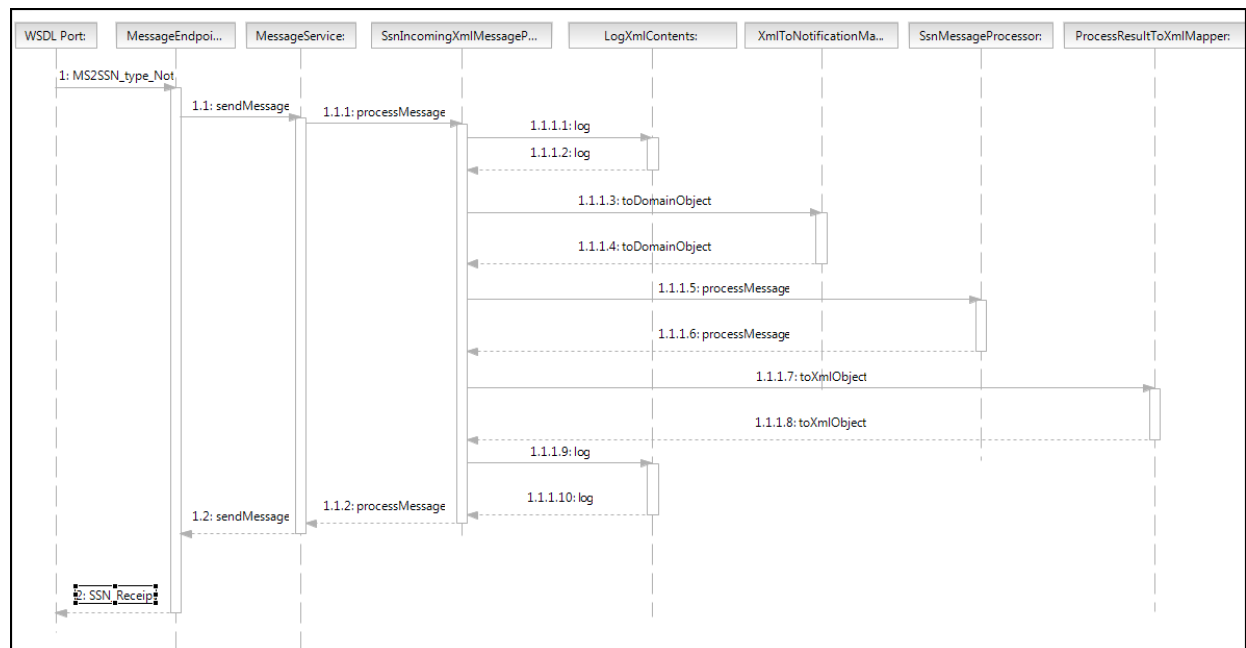


Figure 4-6 Notification / Request / Response Messages

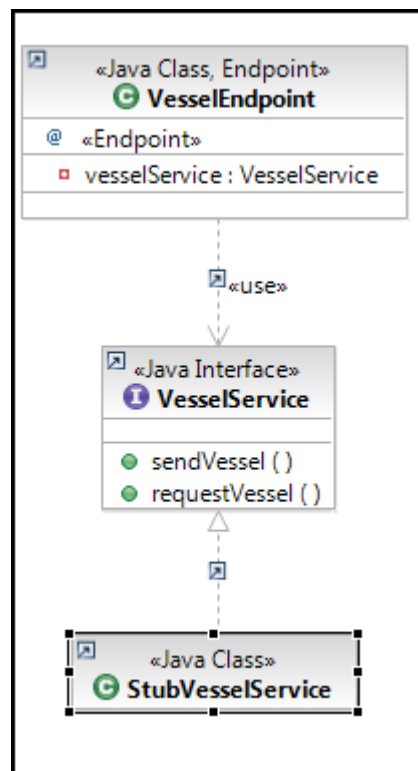
## Sequence Diagram : EIS Incoming XML Message Processor

### Sequence Diagram : EIS Incoming XML Message Processor



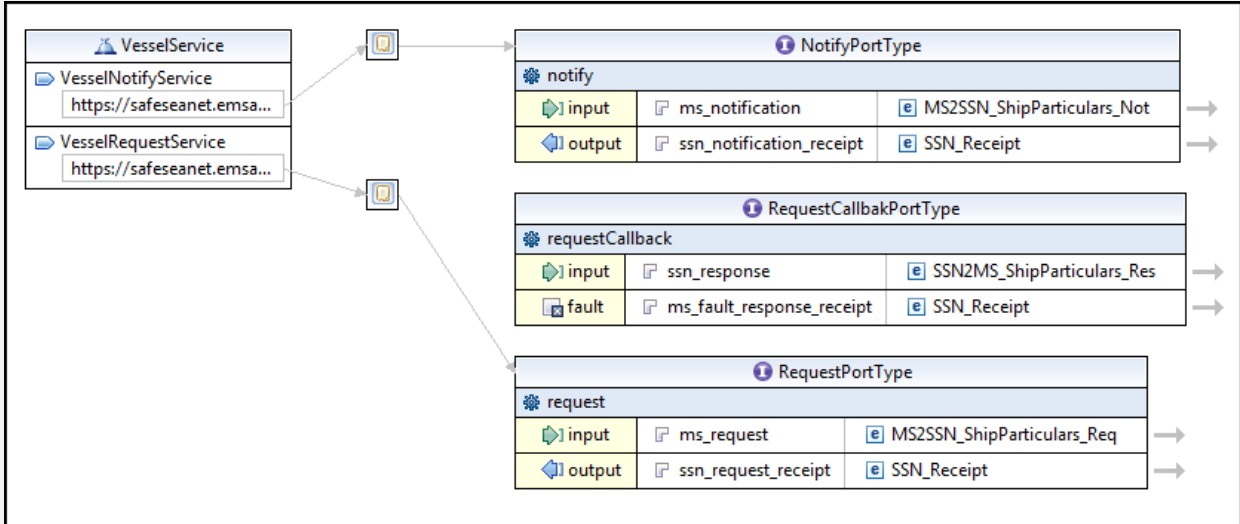
#### 4.4.2.2 WebService: VesselService

### Class Diagram : VesselService



<b>Class</b>	<b>VesselEndpoint</b>
	The payload endpoint handles the incoming SOAP messages. It uses

Class Diagram : VesselService	
	the VesselService to operate. It returns the ProcessResult (Ssn_Receipt) message as response.
<b>Interface</b>	<b>VesselService</b> Provides the business service interface for the incoming messages processing.
<b>Class</b>	<b>StubVesselService</b> The stub implementation of VesselService; it actually delegates the processing to the xml-protocol module processors. The ssn-xmlprotocol-core module uses internally the ssn-xmlprotocol-mapping in order to convert JAXB elements to specific SSN domain objects. Then ssn-xmlprotocol-core delegates the processing to the ssn-core via the ssn-core-ejb.

Service Contract : vessel.wSDL	
 <p>The diagram illustrates the WSDL for VesselService. On the left, a service named 'VesselService' is shown with two endpoints: 'VesselNotifyService' and 'VesselRequestService', both pointing to 'https://safeseanet.emsa...'. These endpoints are connected to three port types on the right: 'NotifyPortType', 'RequestCallbackPortType', and 'RequestPortType'. Each port type has a specific message and associated input/output data types. 'NotifyPortType' has a 'notify' message with inputs 'ms_notification' and 'MS2SSN_ShipParticulars_Not' and outputs 'ssn_notification_receipt' and 'SSN_Receipt'. 'RequestCallbackPortType' has a 'requestCallback' message with inputs 'ssn_response' and 'SSN2MS_ShipParticulars_Res' and outputs 'ms_fault_response_receipt' and 'SSN_Receipt'. 'RequestPortType' has a 'request' message with inputs 'ms_request' and 'MS2SSN_ShipParticulars_Req' and outputs 'ssn_request_receipt' and 'SSN_Receipt'.</p>	
<b>WSDL</b>	<b>Vessel.wSDL</b> Example of service contract WSDL file; it is based on ShipParticulars.xsd schema (data contract).
<b>Data contract</b>	<b>ShipParticulars.xsd</b>

## Vessel Messages

The Figure 4-7 shows the messages (SOAP over HTTP) exchanges between the Data Requestors and EIS Service Provider (Web Services) related to the Vessel (Ship Particulars) Domain as well as the relationship of these messages with the domain classes. More specifically,

- The **ShipParticularsNotification** message is sent by a Member State or system linked to SafeSeaNet (e.g. THETIS, LRIT DC database) in order to **notify** SafeSeaNet new insertions or updates in the ship registry maintained by the data provider. It actually includes the vessel information to be recorded on the EIS database. Vessel Information is represented by the classes consist the vessel package of the domain model (refer section 4.2.2.1.6).
- The **ShipParticularsRequest** message is sent by a MemberState system or THETIS or any other system to be linked to SSN in the future (data requester) in order to **request** the ship particulars of a specific ship registered into the EIS database. It actually consists

of the vessel search criteria. The data requester should provide the **requestCallback** operation where SSN shall send the **ssn\_response(ShipParticularsResponse)** asynchronously.

- The **ShipParticularsResponse** message is the response sent by SafeSeaNet to a MemberState system or EMSA system (e.g. THETIS) any other future system to be connected to SSN requesting ship particulars (**ShipParticularsRequest**). It consists of a list of vessels satisfies the applied search criteria submitted by the corresponding request.

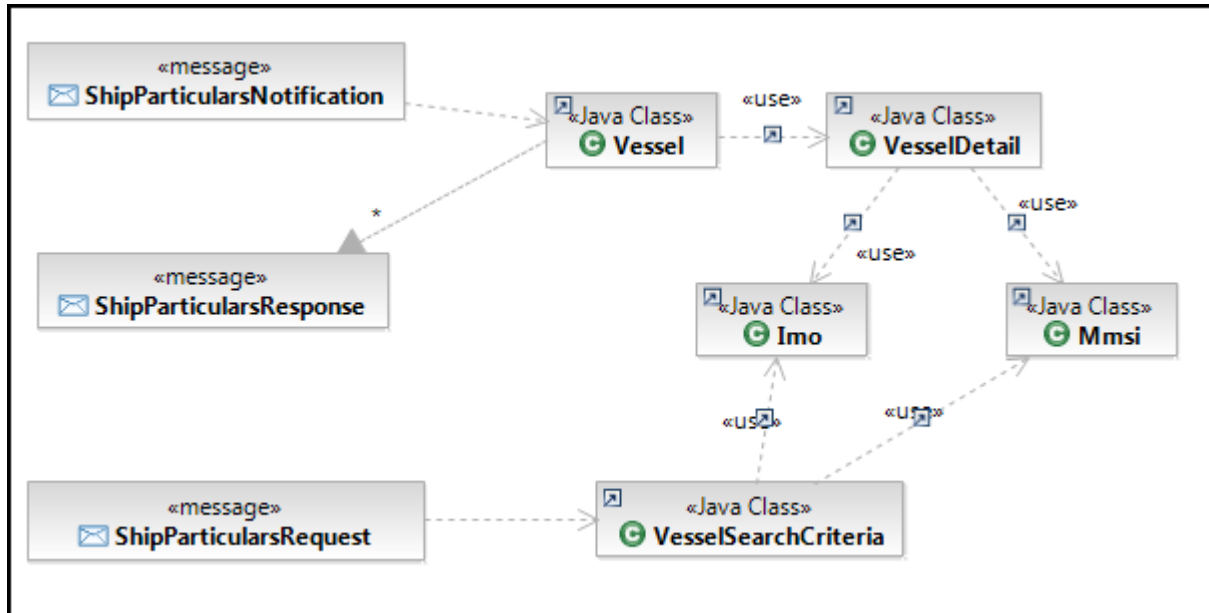
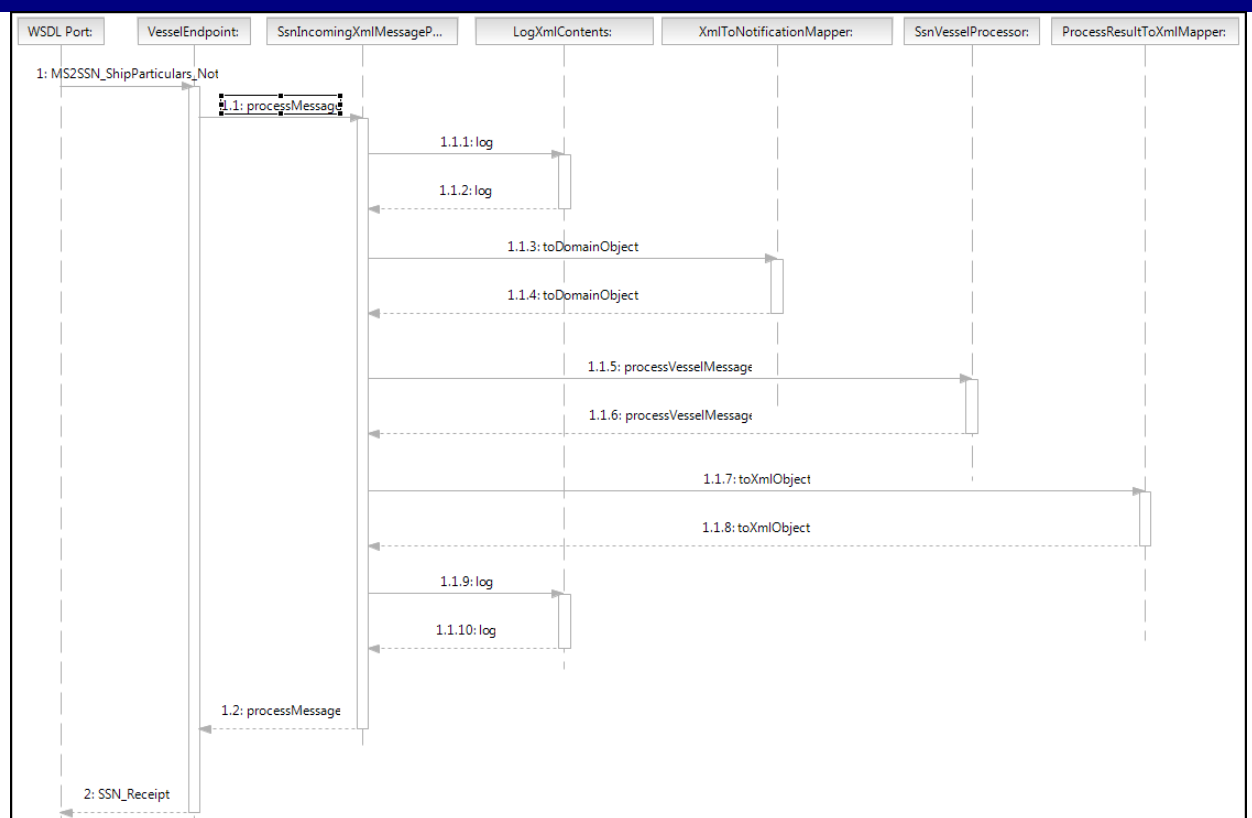


Figure 4-7 Ship Particulars Messages

#### Sequence Diagram : EIS Incoming XML Vessel Processor



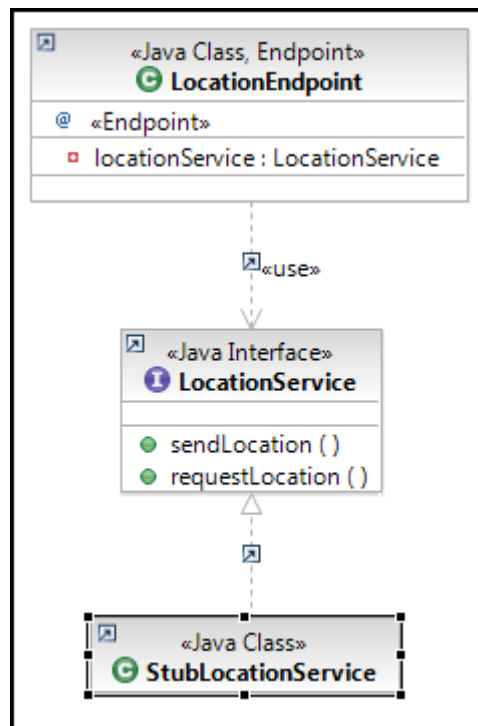
### Sequence Diagram : EIS Incoming XML Vessel Processor



#### 4.4.2.2.3 WebService: LocationService

### Class Diagram : LocationService

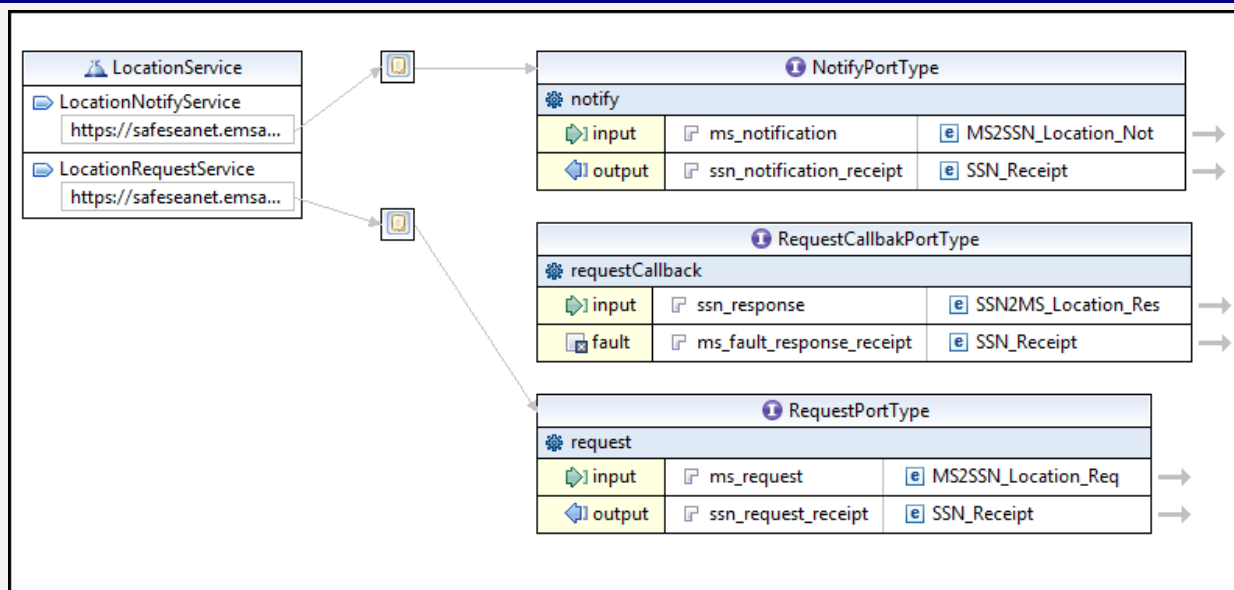
## Class Diagram : LocationService



<b>Class</b>	<b>LocationEndpoint</b> The payload endpoint handles the incoming SOAP messages. It uses the LocationService to operate. It returns the ProcessResult (Ssn_Receipt) message as response.
<b>Interface</b>	<b>LocationService</b> Provides the business service interface for the incoming messages processing.
<b>Class</b>	<b>StubLocationService</b> The stub implementation of LocationService; it actually delegates the processing to the xml-protocol module processors. The ssn-xmlprotocol-core module uses internally the ssn-xmlprotocol-mapping in order to convert JAXB elements to specific SSN domain objects. Then ssn-xmlprotocol-core delegates the processing to the ssn-core via the ssn-core-ejb.

## Service Contract : location.wsdl

## Service Contract : location.wsdl



## WSDL

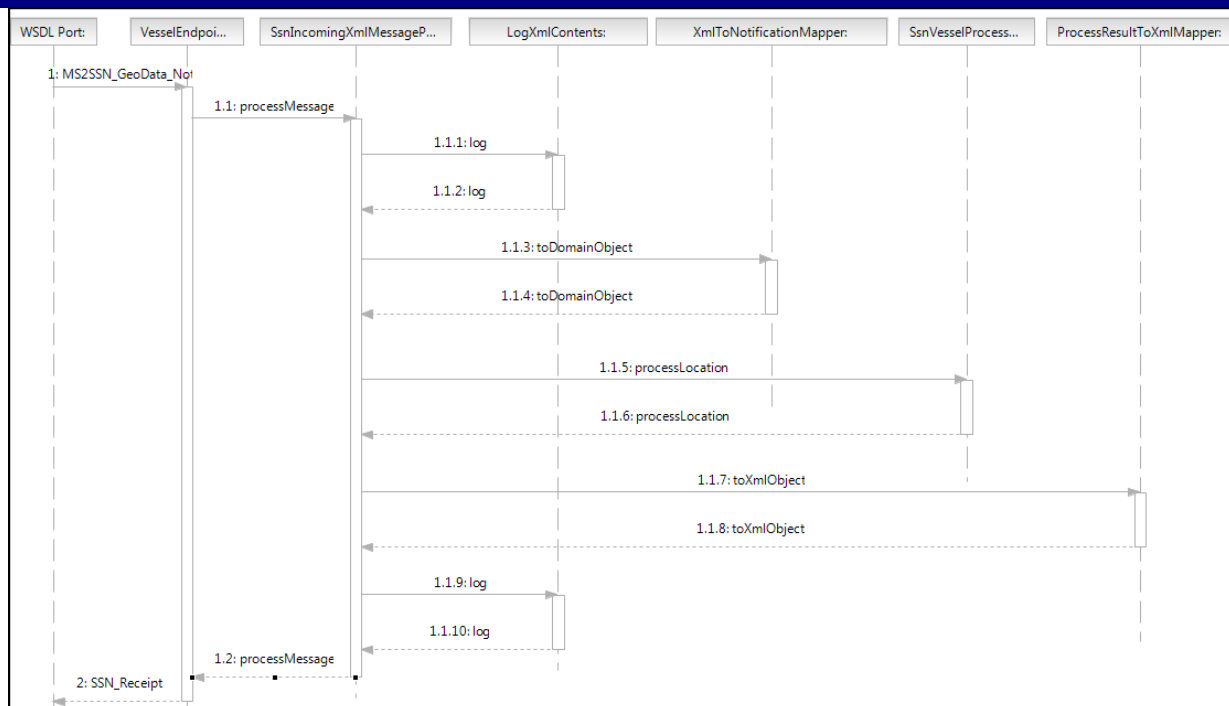
### location.wsdl

Example of service contract WSDL file; it is based on SSN\_Location\_Exchange.xsd schema (data contract).

## Data contract

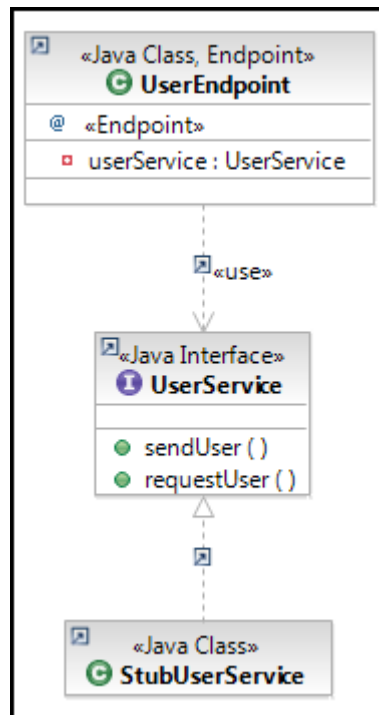
### SSN\_Location\_Exchange.xsd

## Sequence Diagram : EIS Incoming XML Location Processor



#### 4.4.2.2.4 WebService: UserService

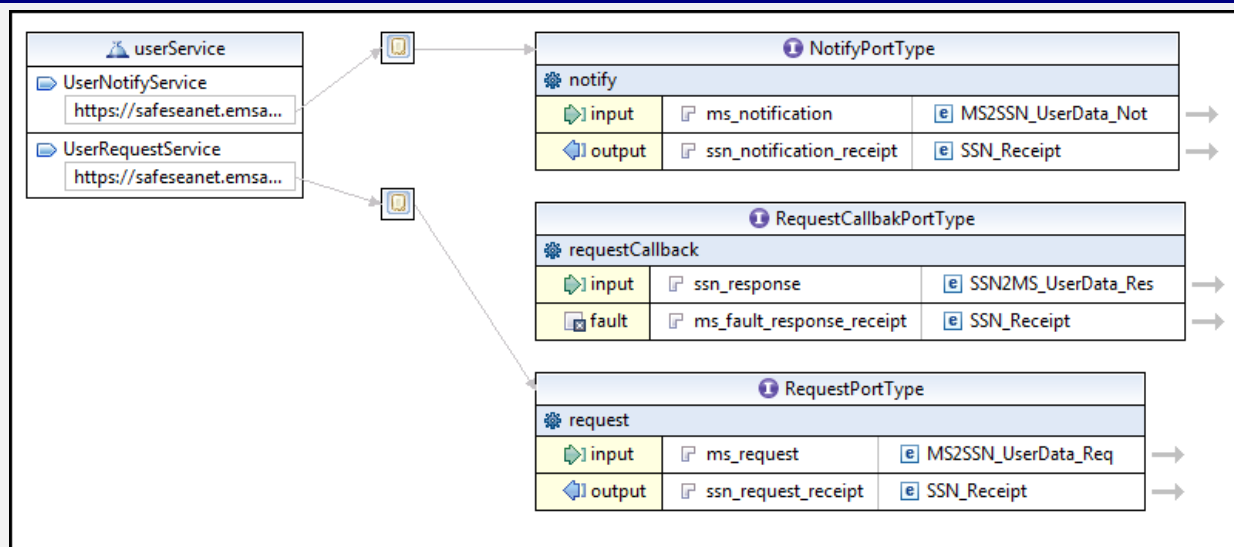
##### Class Diagram : UserService



<b>Class</b>	<b>UserEndpoint</b> The payload endpoint handles the incoming SOAP messages. It uses the <b>UserService</b> to operate. It returns the <b>ProcessResult</b> ( <b>Ssn_Receipt</b> ) message as response.
<b>Interface</b>	<b>UserService</b> Provides the business service interface for the incoming messages processing.
<b>Class</b>	<b>StubUserService</b> The stub implementation of <b>UserService</b> ; it actually delegates the processing to the xml-protocol module processors. The <b>ssn-xmlprotocol-core</b> module uses internally the <b>ssn-xmlprotocol-mapping</b> in order to convert JAXB elements to specific SSN domain objects. Then <b>ssn-xmlprotocol-core</b> delegates the processing to the <b>ssn-core</b> via the <b>ssn-core-ejb</b> .

##### Service Contract : user.wsdl

## Service Contract : user.wsdl



### WSDL

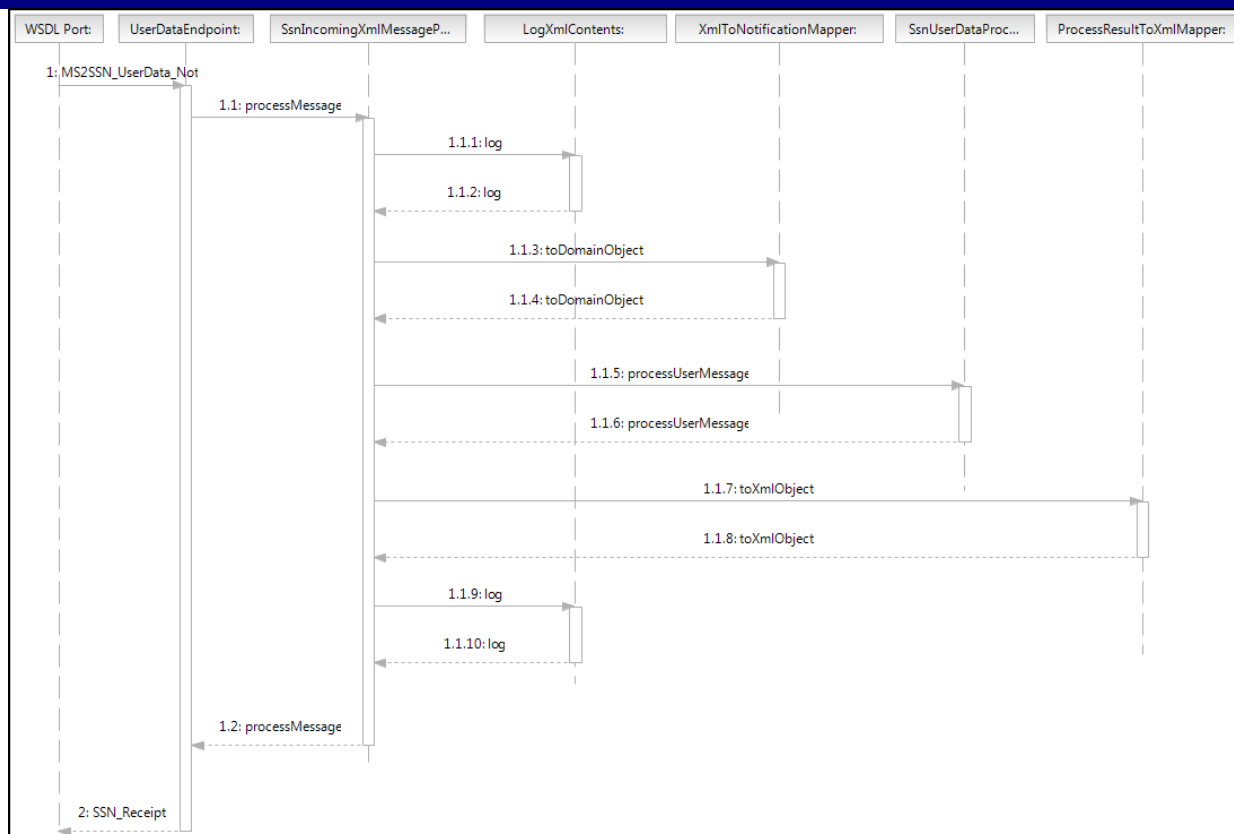
#### user.wsdl

Example of service contract WSDL file; it is based on `SSN_UserData_Exchange.xsd` schema (data contract).

### Data contract

#### SSN\_UserData\_Exchange.xsd

## Sequence Diagram : EIS Incoming XML User Processor

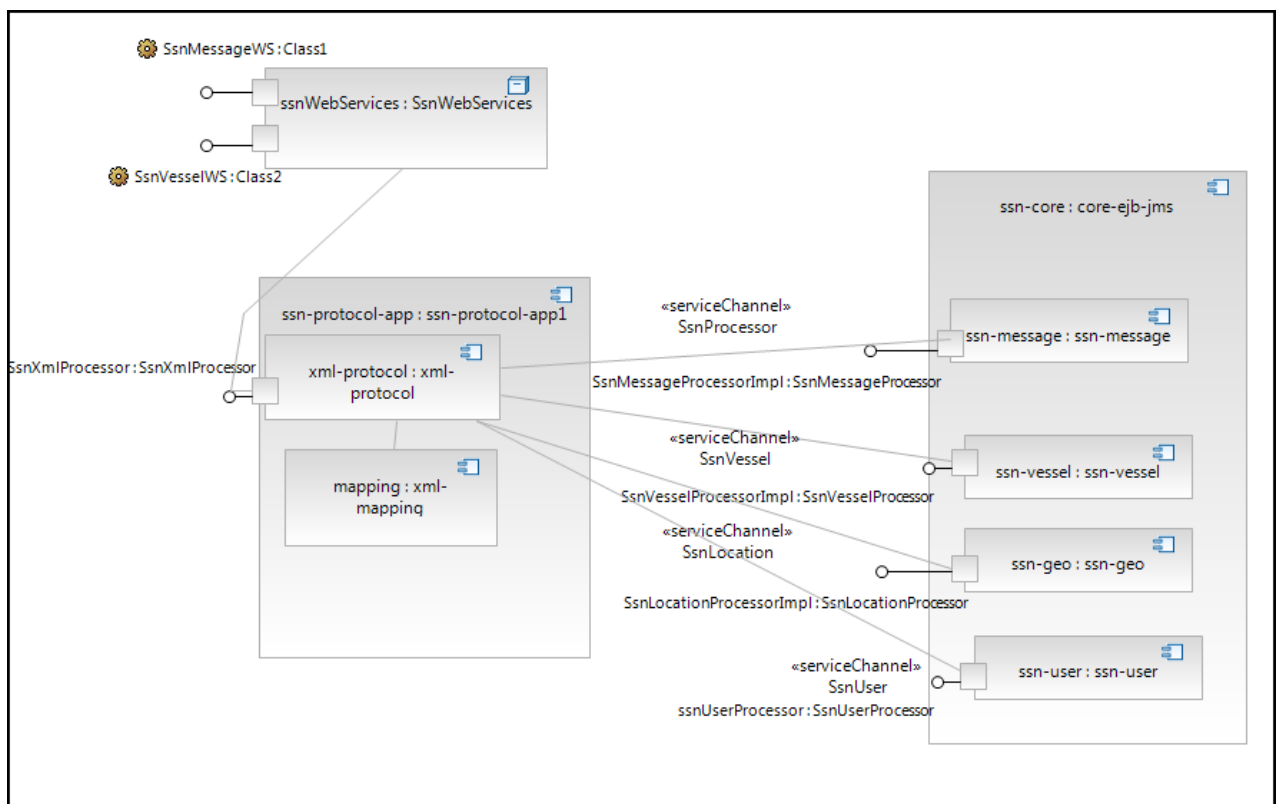


### 4.4.3 Interfaces between the *ssn-xmlprotocol-app* and the *ssn-core-app*

In general, the ***ssn-core-app*** application, accepts calls for processing messages - both synchronously and asynchronously - via the remote stateless session EJB ***SsnMessageProcessorBean***, which is located at package ***ssn.message.processor.ejb*** and actually is a proxy of the class ***SsnMessageProcessorImpl*** - located under the package ***ssn.message.processor***.

The ***SsnMessageProcessorImpl*** implements a single method:

*processMessage(Message message) : ProcessResult*



**Figure 4-8** Interfaces between the *ssn-xmlprotocol-app* and the *ssn-core-app*.

Based on the type of the message the ***SsnMessageProcessorBean*** bean:

- Either calls synchronously the ***NotificationManager*** in case the message is a notification (MS2SSN\_<SSN\_type>\_Not)
- Or sends the message to the ***ssn-core-app***'s incoming messages queue in order the message to be processed asynchronously, if the message is one of the following:
  - InformationRequest (MS2SSN\_xxx\_Req),
  - AdditionalInformationReply (MS2SSN\_xxx\_Res)

The ***ssn-xmlprotocol-app*** uses the ***SsnMessageProcessor*** EJB to route incoming messages to the ***ssn-core-app*** application.

The **ssn-core-app** applications sends asynchronous messages to its clients (ssn-xml-protocol-app, and management console in our case) using its outgoing queue. Each outgoing message has an attribute (attribute protocol of class Message), which denotes the target client application.

The **ssn-xmlprotocol-app** has a permanent listener on the outgoing queue – implemented as the message driven bean **OutgoingMessageProcessorMdb**, located under package **ssn.xmlprotocol.message.processor.ejb**. The listener filters the outgoing queue and consumes only messages that target the **ssn-xmlprotocol-app**.

The types of the messages that are accepted are:

- AdditionalInformationRequest (SSN2MS\_<type>\_Req), or
- InformationReply (SSN2MS\_<type>\_Res).

Each message is converted into xml/soap and posted to the appropriate MS via http protocol.

It should be noted that the system tries to re-deliver a message in case that the http protocol (http client post method) fails - return code  $\geq 400$ . The number of retries is defined by the SSN\_MAX\_RETRY\_COUNT application parameter.

#### 4.4.4 Security on the ssn-xmlprotocol-app

The ssn-xmlprotocol-app application does not have its own security layer. It relies on:

- The system level security provided by the runtime environment (SSL and Client Certificate)
- The application level security offered by ssn-core-app.

### 4.5 SSN-Central Database (CD)

This section provides an overview of the required changes on SSN-EIS System for the management of the SSN entities on the Central Authorities, Locations and Ship Database and the corresponding reporting and consolidation mechanisms.

The SSN system is identified by the following entities owned by the "SSN Resources" Business System

- Users; they are classified to authorities and persons
- Locations
- Countries / Areas (out of the scope of this document)
- Vessels

Thus, SSN system provides a number of services that enable the management (*CRUD*) of these entities as shown in Figure 4-9.

An important upgrade of the aforementioned entities is the identification of the registry where the instance of entity is / to be stored. The type of *registry* attribute is an enumeration with values **Operational**, **Central**.

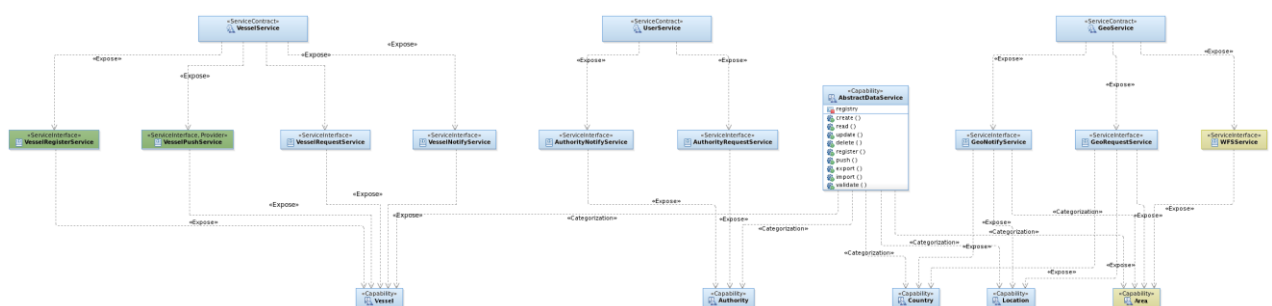


Figure 4-9 SSN CD Services provided operations.



1. **EIS management consoles UI** components provide web interface for the management of the aforementioned entities (Presentation Layer). More specifically,
  - a. **ssn-web-common** component is a library that provides
    - the spring framework functionality as spring web MVC, spring web flow and spring security;
    - the JSF framework functionality;
    - the [RichFaces](#) JSF implementation;
    - ESAPI functionality.

- b. **ssn-vessels-console UI** component that provides web interface for Vessel management. The packaging of this component exposes two deployment artifacts,
  - one for the management of the Operational Vessel Registry; it is actually the isolation of the Vessel management provided by the current version of ssn-admin-console UI component (backward compatibility), and
  - a second for the management of the Base Vessel Registry (Reference Vessel Database);
- c. **ssn-users-console UI** component that provides web interface for User management. The packaging of this component exposes two deployment artifacts,



- one for the management of the Operational Parties Registry; it is actually the isolation of the User management provided by the current version of ssn-admin-console UI component (backward compatibility), and
- a second for the management of the Central Organisation Database (COD);
- d. **ssn-geo-console UI** component that provides web interface for Countries / Locations / Areas management. The packaging of this component exposes two deployment artifacts,
  - one for the management of the Operation Registry; it is actually the isolation of the Countries / Locations / Areas management provided by the current version of ssn-admin-console UI component (backward compatibility), and
  - a second for the management of the Base Central Locations Database (CLD);

The authentication of the aforementioned UI components shall be done via the SSN SSO (IdM).

The communication mechanism between the UI components and the SSN EIS Business services shall not be changed; it shall be based on EJB.

2. **SSN EIS Services** subsystem (Service Layer) exposes the EIS Business functionality as
  - a. Web Services (SOAP over HTTP) and
  - b. EJB 3.0 (stateless session beans).

More specifically,

- a. **ssn-ws-support** component is a library that provides the spring-ws functionality and it is used by all the following components expose their functionality as Web Services;
  - b. **ssn-vessel-service** exposes the EIS Vessel functionality. The Web Services communication is based on the EIS Vessel Services (vesselservice.wsdl) and the SSN Ship Particulars XML schema (SSN\_Ship\_Particulars\_Exchange.xsd).
  - c. **ssn-user-service** exposes the EIS Authorities functionality. The Web Services communication is based on the EIS Authority Services (authorityservice.wsdl) and the SSN Authority XML schema (SSN\_Authority\_Exchange.xsd).
  - d. **ssn-geo-service** exposes the EIS Locations functionality. The Web Services communication is based on the EIS Location Services (locationservice.wsdl) and the SSN Location XML schema (SSN\_Location\_Exchange.xsd).
3. **ssn-core-app** subsystem implements the SSN EIS Business functionality (Business Layer). This subsystem exposes the aforementioned **SSN EIS Services**. The existing version shall be upgraded to provide the required CD functionality.

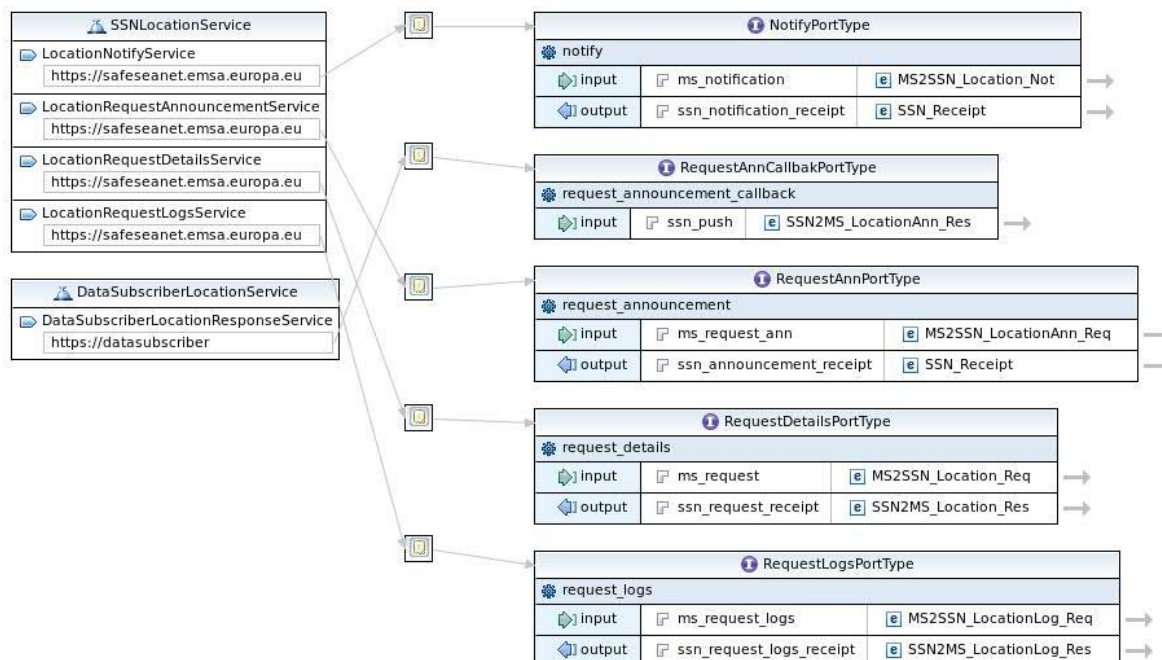
**Data Services;** ssn-dao module, included in the ssn-core-app subsystem, implements the EIS Data Services; it shall be upgraded to take into account the aforementioned *registry* attribute of the SSN EIS entities so to update accordingly the corresponding registry (Operational / Base). The search criteria classes shall be also updated with *registry* attribute for the information retrieval from the corresponding registry.
4. **GIS Server** subsystem represents the EMSA system provides [WFS](#) service (for future implementation); this node is out of the scope of the document.
5. **IdM** subsystem represents the EMSA Identity Management (IdM) system [**Error! Reference source not found.**] provides the user provisioning service; SSN SSO functionality. This node is out of the scope of the document.

The SSN Central Database provides an electronic interface for EIS entities management by Member States and other EMSA systems (External Systems). This section describes this communication provided by the SSN CD Web Services.

## 4.5.1 Location Service

The SSN Location Web Service defines the CLD data exchange.

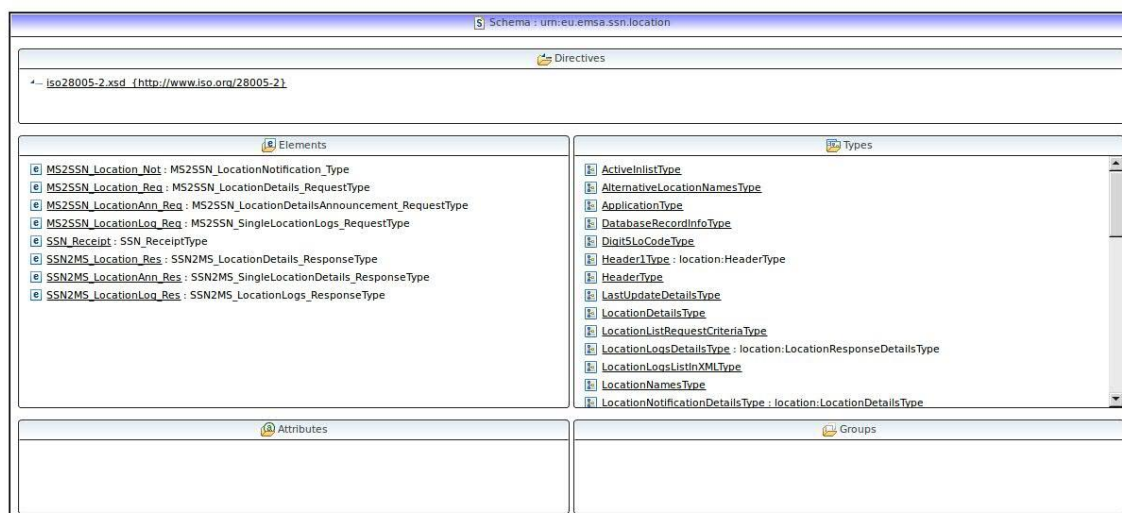
### Service Contract : locationservice.wsdl



### WSDL

#### locationservice.wsdl

Example of service contract WSDL file; it is based on SSN\_Location\_Exchange.xsd schema (data contract).



### Data contract

#### SSN\_Location\_Exchange.xsd

### Location Messages

The following messages (SOAP over HTTP) implements the CLD data exchange. More specifically,

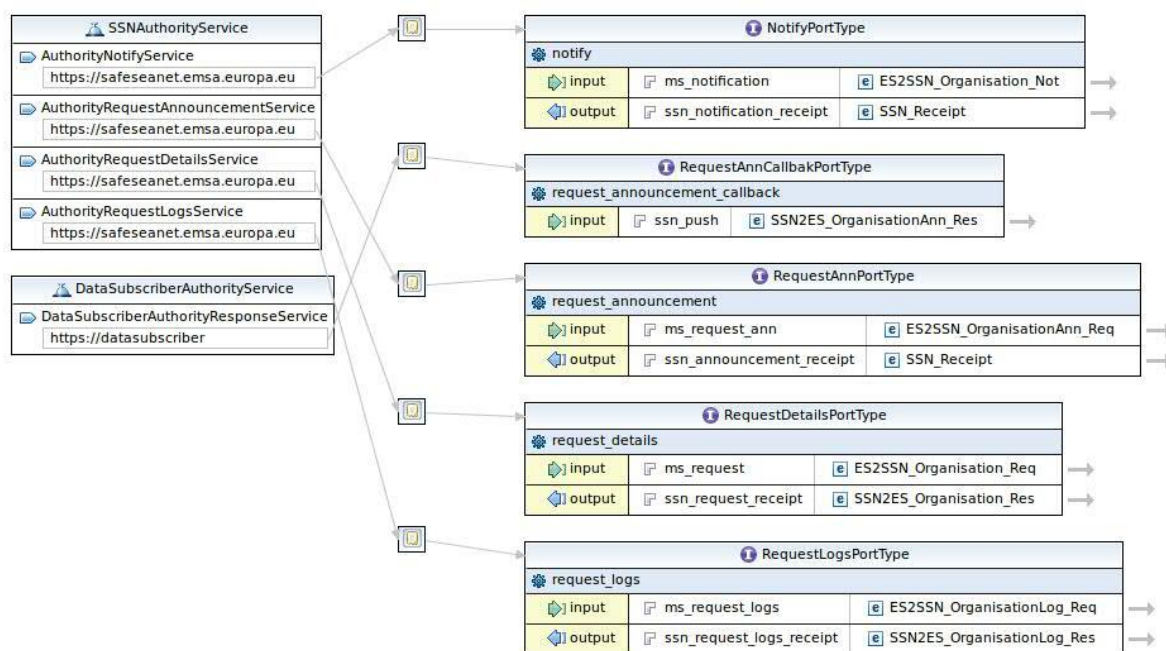
- The **RequestLocationDetails (MS2SSN\_Location\_Req)** message is sent by a Member State system or any other system linked to SSN (data requestor) in order to **request (ms\_request)** the location details (for a unique location or list of locations). This is a synchronous service implemented by exchanging messages between the data requester and the SafeSeaNet system.
- The **ResponseLocationDetails (SSN2MS\_Location\_Res)** message is sent by the SafeSeaNet system to the Member State system or any other system linked to SSN (data requestor) as synchronous response (**ssn\_request\_receipt**) including the location information related to the criteria set of the aforementioned **RequestLocationDetails** message.
- The **RequestLocationLogs (MS2SSN\_LocationLog\_Req)** message is sent by a Member State system or any other system linked to SSN (data requestor) in order to **request (ms\_request\_logs)** the logs for a location. This is a synchronous service implemented by exchanging messages between the data requester and the SafeSeaNet system.
- The **ResponseLocationLogs (SSN2MS\_LocationLog\_Res)** message is sent by the SafeSeaNet system to the Member State system or any other system linked to SSN (data requestor) as synchronous response (**ssn\_request\_logs\_receipt**) including the location logs information related to the criteria set of the aforementioned **RequestLocationLogs** message.
- The **RequestAnnouncement (MS2SSN\_LocationAnn\_Req)** message is sent by a Member State system or any other system linked to SSN (data subscriber) in order to **subscribe (ms\_request\_ann) for CLD changes announcement**. Such service is implemented by exchanging different messages between the data subscriber and the SafeSeaNet system. The data subscriber should provide the **request\_ann\_callback** operation where SSN shall send the **AnnouncementLocation (SSN2MS\_LocationAnn\_Res)** message asynchronously. The **AnnouncementLocation** message is generated by SSN system and delivered to data subscribers; SafeSeaNet system acts as webservice client on this operation (**request\_announcement\_callback**).
- The **AnnouncementLocation (SSN2MS\_LocationAnn\_Res)** message is sent by the SafeSeaNet system to the Member State system or any other system linked to SSN (data subscriber) in order to **deliver (ssn\_push) the CLD changes via the aforementioned message RequestAnnouncement**. This service (**DataSubscriberLocationService**) is implemented by the data subscriber.
- The **NotificationLocation message (MS2SSN\_Location\_Not)** message is sent by a Member State or system linked to SafeSeaNet (acting as Data Provider) in order to **notify** SafeSeaNet for the creation/update and/or de-activation of a single location within an application.

#### 4.5.2 Authority Service

The SSN Authority Web Service defines the COD data exchange.

**Service Contract : authorityservice.wsdl**

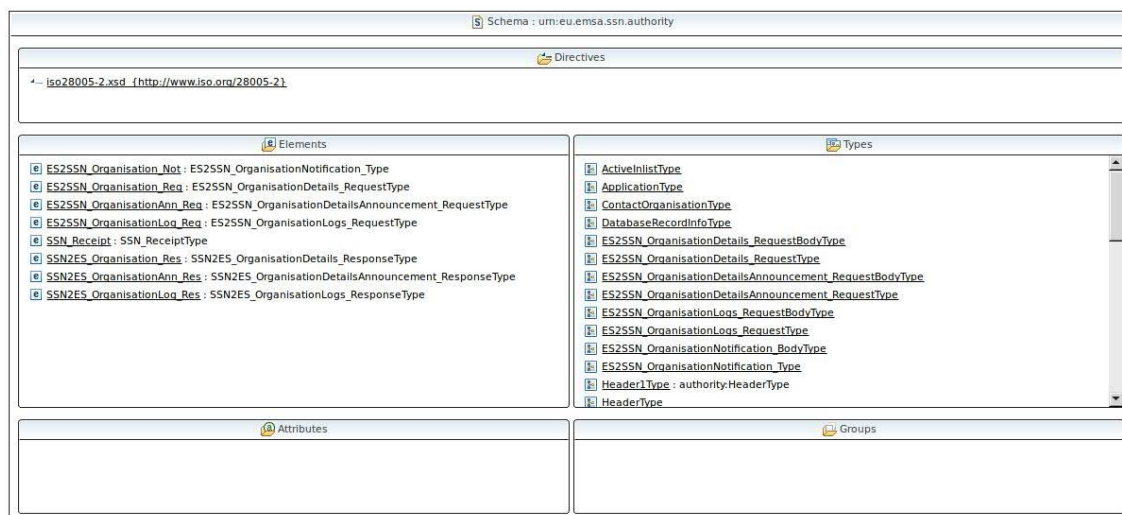
## Service Contract : authorityservice.wsdl



## WSDL

### authorityservice.wsdl

Example of service contract WSDL file; it is based on SSN\_Organisation\_Exchange.xsd schema (data contract).



## Data contract

### SSN\_Organisation\_Exchange.xsd

## Authority Messages

The following messages (SOAP over HTTP) implements the COD data exchange. More specifically,

- The **RequestAuthorityDetails (ES2SSN\_Organisation\_Req)** message is sent by a Member State system or any other system linked to SSN (data requestor) in order to **request (ms\_request)** the authority details (for a unique authority or list of authorities). This is a synchronous service implemented by exchanging messages between the data requester and the SafeSeaNet system.
- The **ResponseAuthorityDetails (SSN2ES\_Organisation\_Res)** message is sent by the SafeSeaNet system to the Member State system or any other system linked to SSN (data requestor) as synchronous response (**ssn\_request\_receipt**) including the authority information related to the criteria set of the aforementioned **RequestAuthorityDetails** message.
- The **RequestAuthorityLogs (ES2SSN\_OrganisationLog\_Req)** message is sent by a Member State system or any other system linked to SSN (data requestor) in order to **request (ms\_request\_logs)** the logs for an authority. This is a synchronous service implemented by exchanging messages between the data requester and the SafeSeaNet system.
- The **ResponseAuthorityLogs (SSN2ES\_OrganisationLog\_Res)** message is sent by the SafeSeaNet system to the Member State system or any other system linked to SSN (data requestor) as synchronous response (**ssn\_request\_logs\_receipt**) including the authority logs information related to the criteria set of the aforementioned **RequestAuthorityLogs** message.
- The **RequestAnnouncement (ES2SSN\_OrganisationAnn\_Req)** message is sent by a Member State system or any other system linked to SSN (data **subscriber**) in order to **subscribe/ unsubscribe (ms\_request\_ann) for COD changes announcement**. Such service is implemented by exchanging different messages between the data **subscriber** and the SafeSeaNet system. The data **subscriber** should provide the **request\_ann\_callback** operation where SSN shall send the **AnnouncementAuthority (SSN2ES\_OrganisationAnn\_Res)** message asynchronously. The **AnnouncementAuthority** message is generated by SSN system and delivered to data subscribers; SafeSeaNet system acts as webservice client on this operation (**request\_announcement\_callback**).
- The **AnnouncementAuthority (SSN2ES\_OrganisationAnn\_Res)** message is sent by the SafeSeaNet system to the Member State system or any other system linked to SSN (data **subscriber**) in order to **deliver (ssn\_push) the COD changes via the aforementioned message RequestAnnouncement**. This service (**DataSubscriberAuthorityService**) is implemented by the data **subscriber**.
- The **NotificationAuthority message (MS2SSN\_Authority\_Not)** message is sent by a Member State or system linked to SafeSeaNet (acting as Data Provider) in order to **notify** SafeSeaNet for the creation / update of an organisation to an application interacting with the base registry.

### 4.5.3 Ship Particulars Service

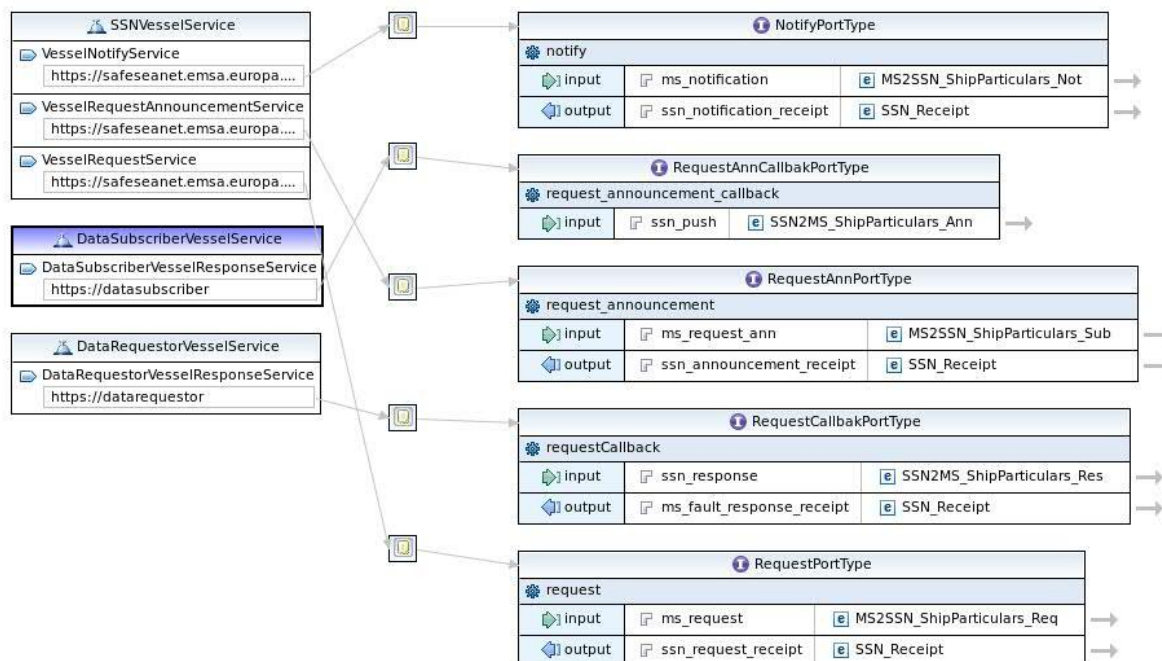
The SSN Ship Particulars Web Service is upgraded to

- "Push" CSD updates to MS subscribers;
- Provide details on the history of changes in the records of the CSD upon request.

**Service Contract : vesselservice.wsdl**



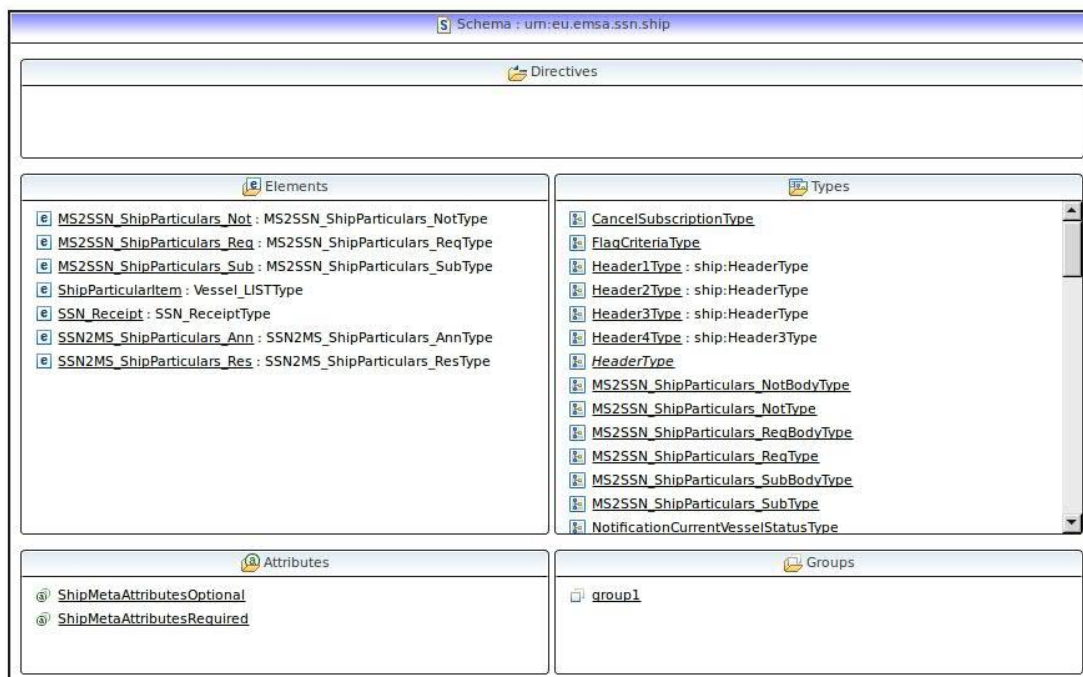
## Service Contract : vessel.service.wsdl



## WSDL

### vessel.service.wsdl

Example of service contract WSDL file; it is based on SSN\_Ship\_Particulars\_Exchange.xsd schema (data contract).



**Service Contract : vesselservice.wsdl**

<b>Data contract</b>	<b>SSN_Ship_Particulars_Exchange.xsd</b>
----------------------	--

### Ship Particulars Messages

The following messages (SOAP over HTTP) implements the aforementioned new functionality. More specifically,

- The **RequestAnnouncement (MS2SSN\_ShipParticulars\_Sub)** message is sent by a Member State system or any other system linked to SSN (data **subscriber**) in order to **subscribe/unsubscribe (ms\_request\_ann) for the ship particulars announcement**. Such service is implemented by exchanging different messages between the data **subscriber** and the SafeSeaNet system. The data **subscriber** should provide the **request\_ann\_callback** operation where SSN shall send the **AnnouncementShipParticulars (SSN2MS\_ShipParticulars\_Ann)** message asynchronously. The **AnnouncementShipParticulars** message is generated by SSN system and delivered to data subscribers; SafeSeaNet system acts as webservice client on this operation (**request\_shipparticulars\_callback**).
- The **AnnouncementShipParticulars (SSN2MS\_ShipParticulars\_Ann)** message is sent by the SafeSeaNet system to the Member State system or any other system linked to SSN (data **subscriber**) in order to **deliver (ssn\_push) the ship particulars information to the registered data subscribers**. This service (**DataSubscriberVesselService**) is implemented by the data **subscriber**.

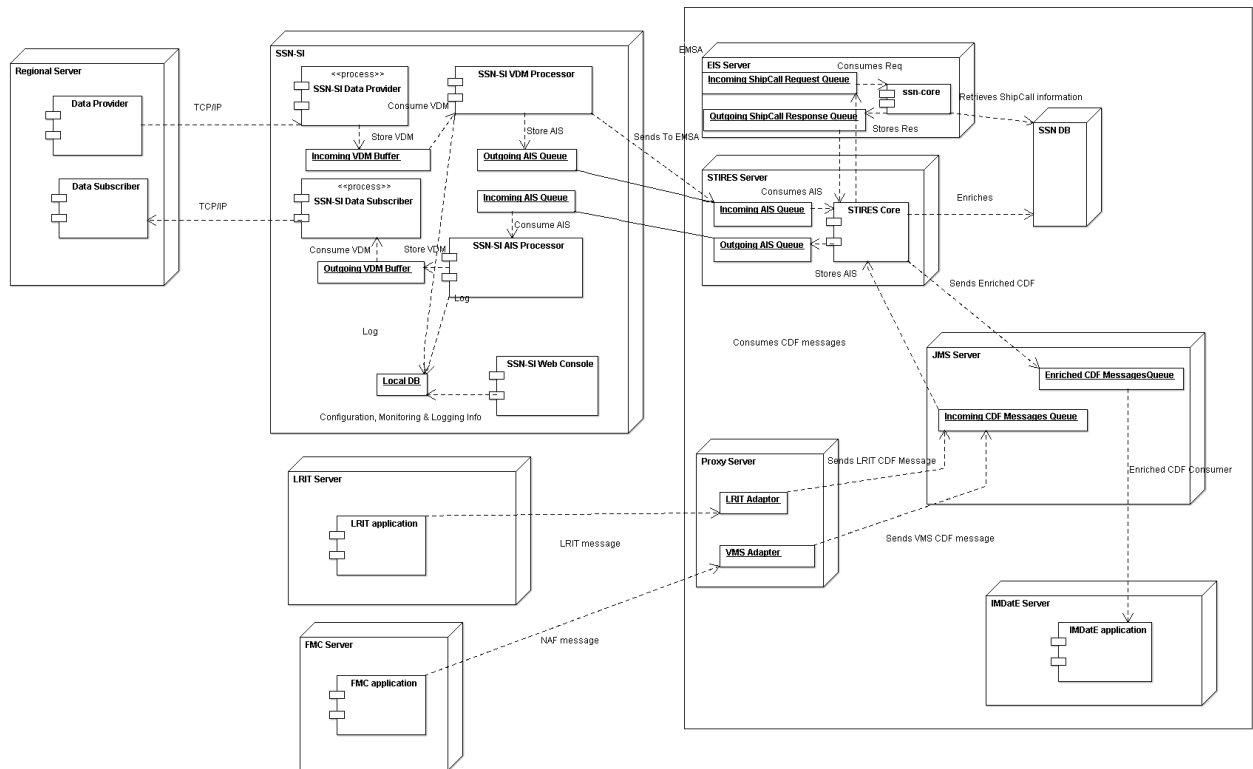
The following messages (SOAP over HTTP) implements the existing functionality;

- The **ShipParticularsNotification** message is sent by a Member State or any other system linked to SSN in order to **notify** SafeSeaNet new insertions or updates in the ship registry maintained by the data provider. It actually includes the vessel information to be recorded on the EIS database.
- The **ShipParticularsRequest** message is sent by a Member State system or any other system linked to SSN (data requester) in order to **request** the ship particulars of a specific ship registered into the EIS database. It actually consists of the vessel search criteria. The data requester should provide the **requestCallback** operation where SSN shall send the **ssn\_response (ShipParticularsResponse)** asynchronously.
- The **ShipParticularsResponse** message is the response sent by SafeSeaNet to a Member State system or EMSA system (e.g. THETIS) any other future system to be connected to SSN requesting ship particulars (**ShipParticularsRequest**). It consists of a list of vessels satisfies the applied search criteria submitted by the corresponding request.

## 4.6 SSN-IMDaTe

This section provides an overview of the SSN-IMDaTe integration diagram. It describes

- the SSN-SI system; to be upgraded to support S-AIS messages;
- the SSN system; to be upgraded to exchange information in CDF format with
- the IMDaTe system.



**Figure 4-11** Connection diagram of SSN IMDatE components exchange AIS messages.

Figure 4-11 depicts the following subsystems exchange AIS messages:

4. **SSN-SI** Proxy has three interfaces that provide:
  - a. "real time" data transfer from the data provider (Regional Server);
  - b. "real time" data transfer to the data subscribers (Regional Server);
  - c. data exchange in JSON format (RESTful services) with SSN.
5. **SSN STIRES** (ref: section 4.7) provides:
  - a. RESTful services for data transfer from/to the other systems (SSN-SI, IMDatE, SSN EIS);
  - b. Transformation services (marshal/unmarshal) of exchanged information;
  - c. PL/SQL stored procedures for the SSN Database interaction (vessel identification and enrichment).
6. **SSN EIS** provides:
  - a. RESTful services for data transfer from/to the other systems (SSN-SI, JMS Server, STIRES);
  - b. Transformation services (marshal/unmarshal) of exchanged information;
  - c. PL/SQL stored procedures for the SSN Database interaction (vessel identification and enrichment, shipcall management).
7. **SSN Proxy Server**<sup>3</sup> provides
  - a. Transformation services; adapters transform position messages (LRIT, VMS) from raw to SPCDF format.
  - b. Services for data transfer to JMS server
8. **SSN JMS Server** provides

<sup>3</sup>The proxy server is actually a module provides transformation services; it can be located on STIRES or JMS Server.



- a. JMS queuing services.
- 9. **IMDatE** provides
  - a. Services for data transfer from/to JMS server;
- 10. LRIT and VMS Data provider systems (out of the scope of this document).

The first application (ssn-si-core-app) materialises the main functionality of SSN-SI at national proxy level, which is the management of AIS position reports (in VDM format) (plus connection configuration settings, etc).

The second application (stires-core-app) materialises the main functionality of SSN STIRES, which is the management of AIS messages (plus connection configuration/authentication settings, monitoring, etc).

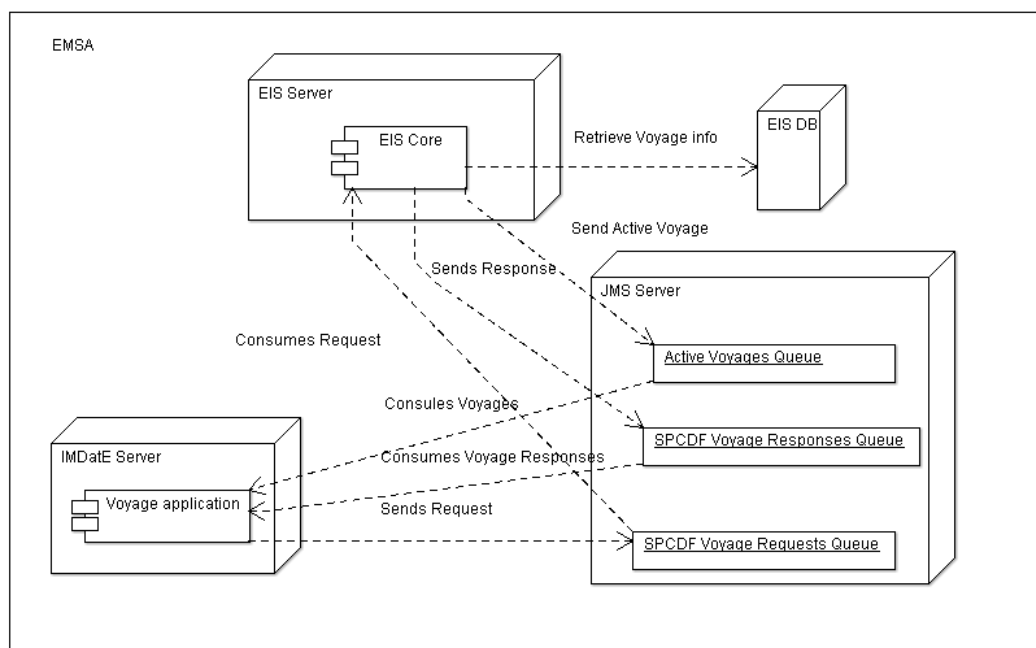
Both applications manage the information of AIS messages in a way that is independent from the channel of communication through which messages are exchanged.

The third application (ssn-core) materialises the main functionality of SSN EIS, which is (for the context of current document) the management of Ship Call Request / Response messages.

The fourth application (Proxy Server) provides transformation services for position messages from raw to SPCDF format so to exchange the position information with SSN in canonical format.

The fifth application (JMS Server) provides JMS queuing services.

The RESTful Services (HTTPS) as exchange endpoints are recommended due to the generic/independent protocol they use; furthermore, the payloads of RESTful services shall be stored on JMS Queues for further processing by IMDatE. OSB supports RESTful Services.



**Figure 4-12 Connection diagram of SSN IMDatE components exchange Voyage messages.**

Figure 4-12 depicts the following subsystems exchange Voyage information:

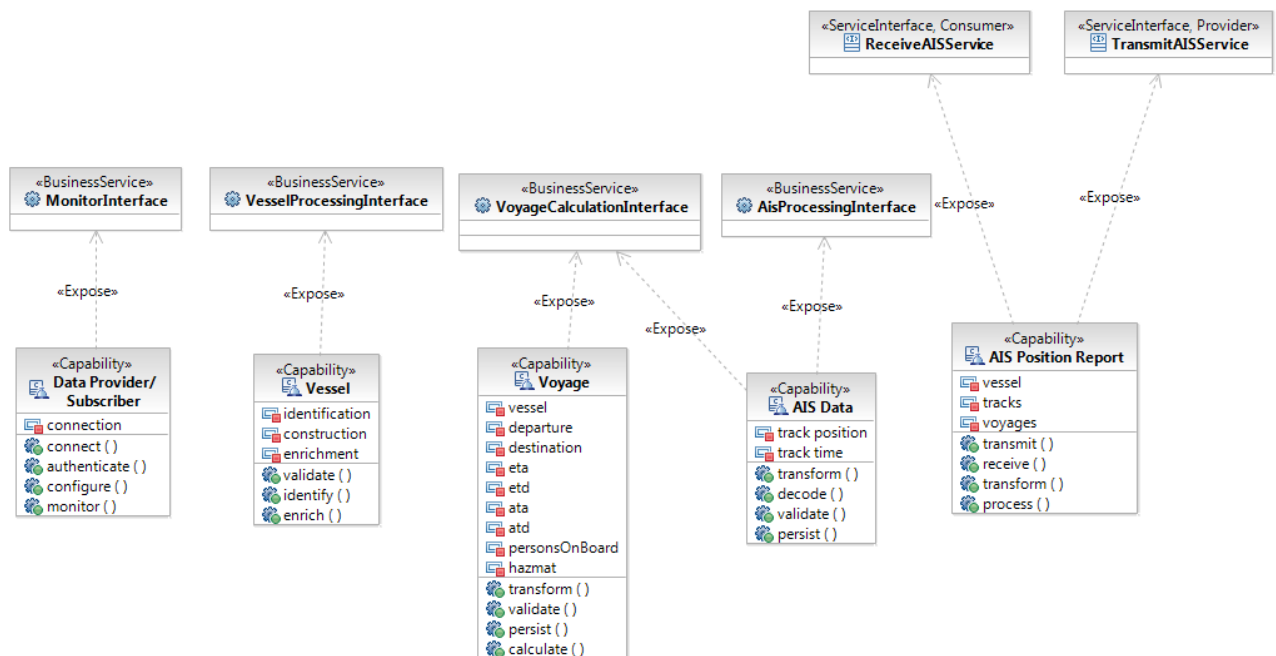
1. **SSN EIS** provides:
  - a. JMS services for data transfer from/to the JMS Server;
  - b. Transformation services (marshal/unmarshal) of exchanged information;
  - c. PL/SQL stored procedures for the SSN Database interaction (voyages retrieval and calculation).
2. SSN JMS Server provides

- a. JMS queuing services.
3. **IMDatE** provides
  - a. Services for data transfer from/to JMS Server.

The SSN-IMDatE system integration requires the following information (entities) to be owned by the "SSN" Business System

- AIS Data/Position Report;
- Vessel Identification and
- Enrichment;
- Voyage Information
- Connection information/configuration (Data Providers/Subscribers).

Thus, SSN system provides services that enable the access to, and update of, these entities as shown in Figure 4-13.



**Figure 4-13 SSN Service provided operations.**

## 4.6.1 Domain module/package

This module/package includes the definition of AIS entities (position reports, vessel, proxy, etc) and it is used by all the AIS sub-systems (sources).

**An XSD (canonical format) shall be created based on the AIS domain named ssnAis.xsd that shall extend/based on the IVEF.xsd and SPCDF.** This schema shall also take into account the ais.xsd that includes the AIS Position Reports.

### 4.6.1.1 UML Class Diagrams

This section covers the architectural significant elements of the design model. It presents the definition of the most significant classes that will implement the requested functionality, organised into packages.

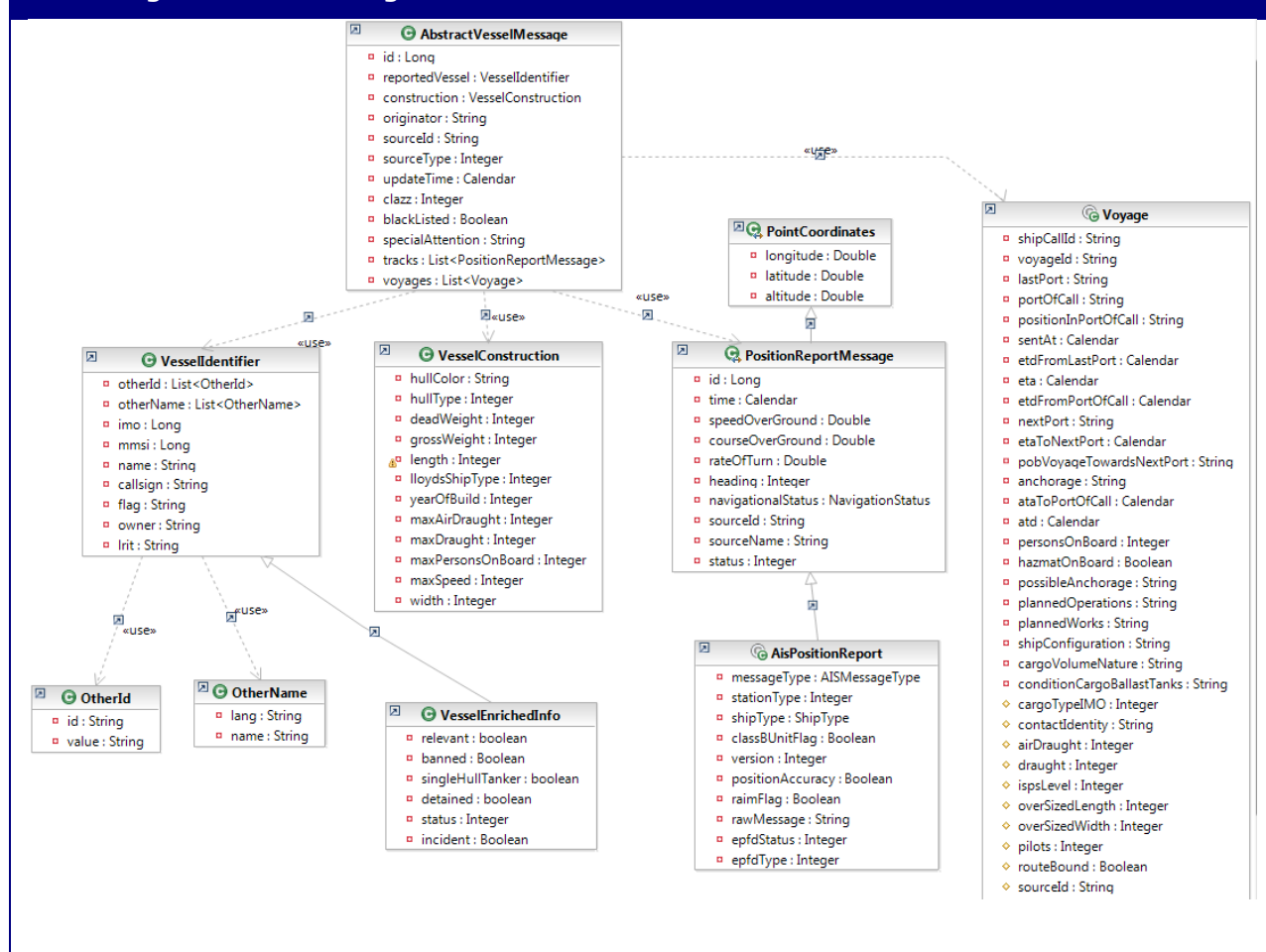
The classes are organised in packages according to the functionality they provide. A package is a general-purpose model element that organizes model elements into groups. Each package

contains a set of classes and interfaces, representing what will become components in the implementation.

#### 4.6.1.2 Module: ssn-domain

##### 4.6.1.2.1 Package: ssn.ais.domain

#### Class Diagram : AIS message



<b>Class</b>	<b><u>VesselIdentification</u></b> This class represents the vessel identification details.
<b>Class</b>	<b><u>VesselEnrichedInfo</u></b> This class extends VesselIdentification class to include the SSN enrichment properties such as banned, relevant, single hull tanker, detained, incident occurrence.
<b>Class</b>	<b><u>VesselConstruction</u></b> This class represents the vessel construction details.
<b>Class</b>	<b><u>AbstractVesselMessage</u></b> The abstract message related to a particular vessel. It includes the following attributes: <ul style="list-style-type: none"> <li>➤ vessel: vessel's identification and (optional) construction</li> <li>➤ updateTime: the message time stamp (in UTC timeframe)</li> <li>➤ source: Organizational source of data for the message that may represent a data service provider, a data management system, an AIS transmission system, etc.</li> <li>➤ tracks: a list of position report messages</li> <li>➤ voyages: a list of voyages information</li> </ul>

Class Diagram : AIS message	
<b>Class</b>	<b><u>PointCoordinates</u></b> This class encapsulates the properties of a point such as latitude, longitude, altitude.
<b>Class</b>	<b><u>PositionReportMessage</u></b> This class extends PointCoordinates class to encapsulate the various dynamic properties of Position report messages at a given point in time and space; it includes SOG, COG, ROT, longitude, latitude, true heading, data source.
<b>Class</b>	<b><u>AisPositionMessage</u></b> This class extends PositionReportMessage class to encapsulate the properties of AIS messages; it includes raim and position accuracy flags, dimension/reference for position of the electronic position fixing device (EPFD) antenna.
<b>Class</b>	<b><u>Voyage</u></b> This class encapsulates the properties of a vessel voyage; it includes departure and destination information, persons on board, hazmat on board, etc.

## 4.7 STIRES Core

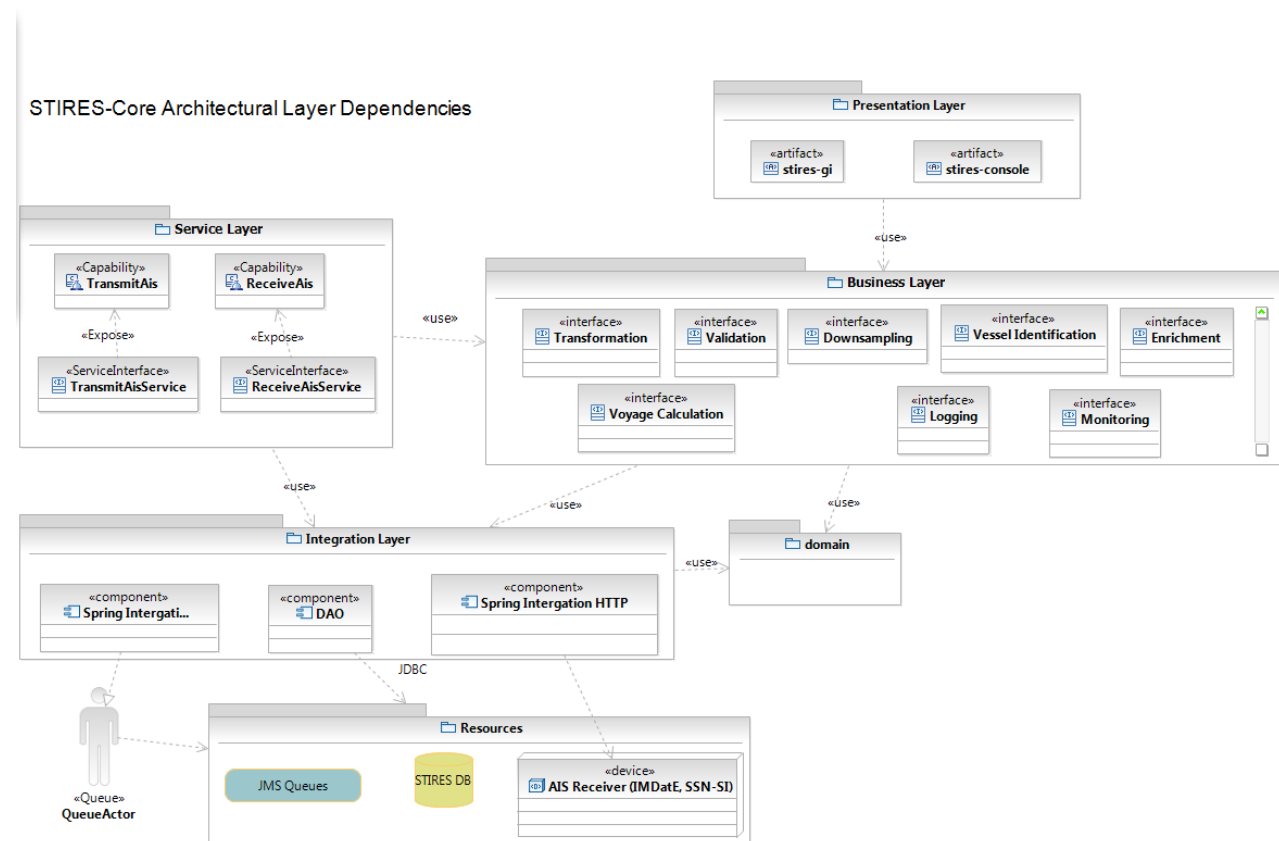
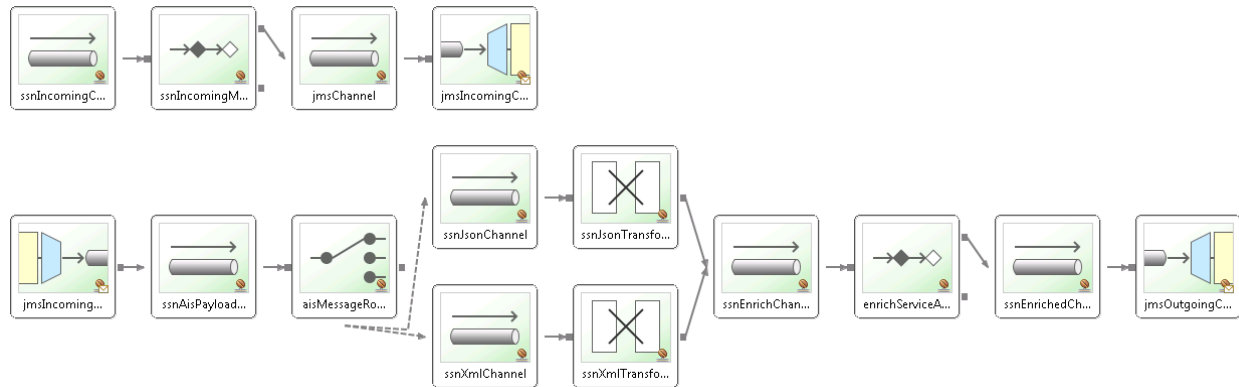


Figure 4-14 Architectural Layer dependencies (Logical view) of STIRES-Core

The stires-core application shall exchange the AIS messages with

- SSN-SI applications over HTTPS (RESTful Services) with JSON payload,
- JMS server over t3 with XML payload

The **incoming AIS messages** (from SSN-SI proxies and IMDatE stored to **Incoming AIS Queue**) shall be processed by a pool of “consumers” (AIS Message Processor/Data Processing Service). Using the Spring Integration module; each AIS message shall be transformed (to POJO), validated, identified, enriched and stored to STIRES DB by the corresponding channels.



**Figure 4-15 Transformation adapters for exchanged AIS messages**

Figure 4-15 shows the usage of Spring Integration transformation adapters so that pre-processing functionality is independent of the format of incoming messages (JSON and XML). Corresponding transformation adapters are used for the enriched outgoing messages according to their destination (JSON for SSN-SI, XML for IMDatE).

The STIRES-Core performs pre-processing using the information stored on the SSN DB and creates AIS messages for the enabled Proxies (and IMDatE that treated as a Proxy) that includes

- Position Reports;
- Enriched Vessel information;
- Voyage information.

The **outgoing AIS messages** (stored to **Outgoing AIS Queue**) shall be processed by a pool of “consumers”. Using the Spring Integration module; each AIS message shall be packed (JSON/XML message) and sent to corresponding Proxies via HTTPS and to the JMS server via t3 when available (the standard redelivery JMS mechanism shall be configured to retry submit the AIS messages for temporarily unavailable proxies).

## 1. Data Services

The DAO module provides CRUD operations to Business Layer; it uses JDBC for the connectivity with STIRES DB.

## 2. Business Layer

This layer provides the Data processing such as the message transformation, validation, identification, enrichment.

## 3. Service Layer

This layer exposes the STIRES-Core functionality as RESTful Services.

**Receive AIS Report capability;** it accepts the AIS Report messages over HTTPS. The AIS Report messages include the Position Reports received from Data Providers (SSN-SI and JMS server).

**Transmit AIS Report capability;** it sends the Enriched AIS messages to Data Subscribers (SSN-SI and JMS server).

## 4.8 SSN-VMS

This section provides an overview of the SSN VMS System. SSN VMS System shall be a new part of current STIRES System; i.e. new modules shall be created to implement the required VMS messages. The SSN VMS System shall use the current SSN GIS Services to provide the functionality concerned with the visualization requirements.

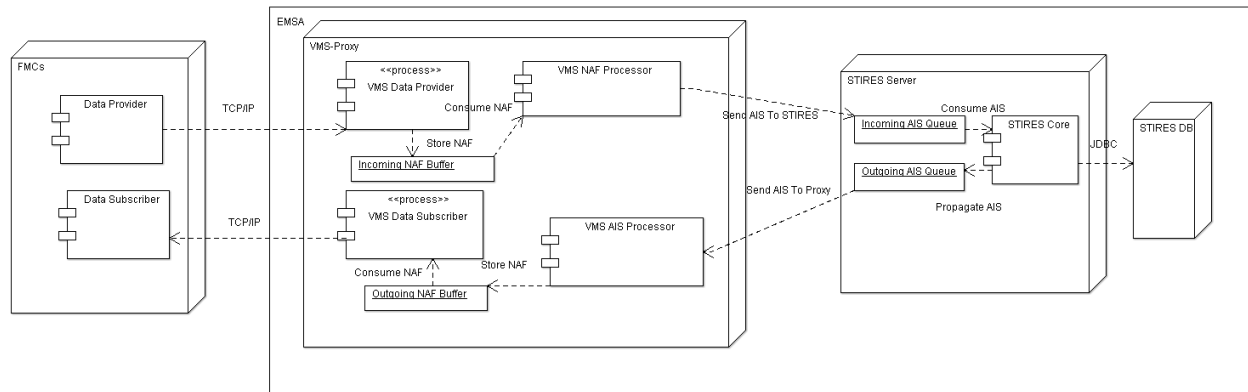


Figure 4-16 Connection diagram of VMS components.

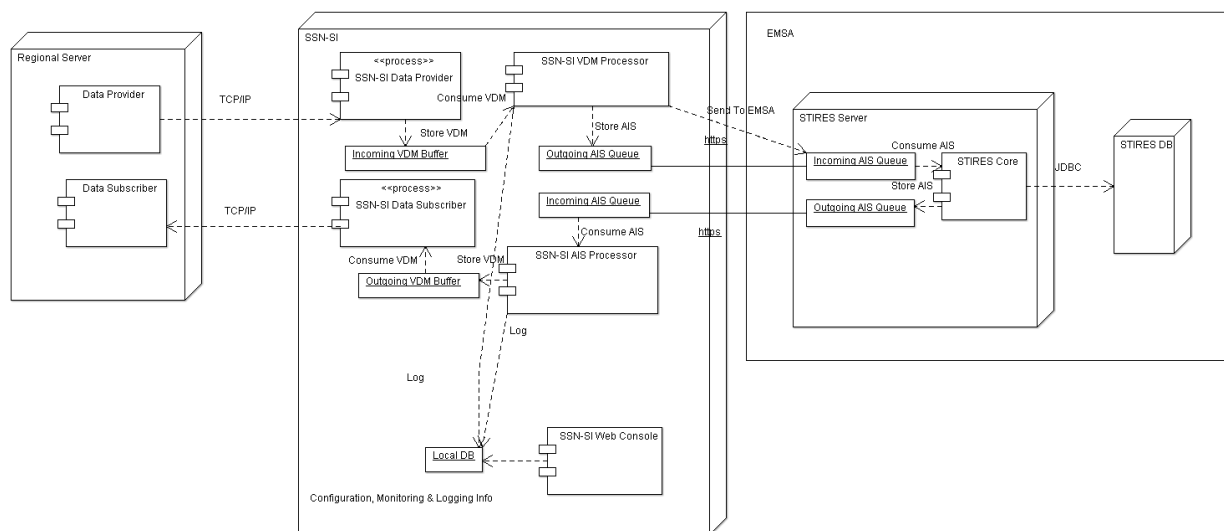


Figure 4-17 Connection diagram of SSN-SI components.

A sovereign element in the design of the VMS System and the upgraded SSN-SI System, as depicted in the diagrams presented in Figure 4-16 and Figure 4-17 respectively, is the division of STIRES system in the following subsystems:

11. VMS Proxy (*ref*: section ) has three interfaces that provide:
  - a. "real time" data transfer from the data provider (FMCs);
  - b. "real time" data transfer to the data subscribers (FMCs);
  - c. data exchange in canonical format (RESTful services) with STIRES Core.
12. SSN-SI Proxy has four interfaces that provide:
  - d. "real time" data transfer from the data provider (Regional Server);
  - e. "real time" data transfer to the data subscribers (Regional Server);
  - f. data exchange in canonical format (RESTful services) with STIRES Core.

- g. storage/retrieval of configuration data (e.g. distinct stream monitoring, filters, authentication, displaying new availability indicators, potential addition of configuration parameters, etc).

13. **STIRES Core** (*ref:section*) provides:

- a. RESTful service for data transfer from/to the VMS-Proxy;
- b. PL/SQL stored procedures for the STIRES Database interaction;

14. **Web Application** includes the following subsystems:

- a. Cartographic Viewer Services; the **SSN GI** shall be also **upgraded** to provide the visualization of VMS functionality.
- b. Human Interface Services; consists in the following modules:

- i. User Manager;

- SSN Admin console shall be used for User definition and task assignment.  
STIRES admin console shall be used for Profiles definition that grant access to STIRES specific entities.

- ii. Proxy Manager;

- The Proxy Management tool shall allow the STIRES administrators to manage STIRES proxies. The module shall allow the following proxy management functions:

- Proxy creation.
    - Proxy settings (enable, disable proxy, change policies and preferences, change countries).

- In order to allow Proxies to receive AIS data, STIRES administrators will manage the proxy's preferences selecting from the following set of information for each proxy:

- Originators;
    - Ship types;
    - Ship flags;
    - Message types;
    - Areas.

- Each proxies will receive or not certain AIS data in base of these settings of preferences.

- Proxies' Policies will define the forwarding rules for AIS messages. The Proxy Management tool shall allow STIRES administrators to enable/disable forwarding rules to one or more countries (receivers).

- iii. System Monitor;

- iv. Data Maintainer.

- c. Track Management Services;
- d. Data Interaction Services;
- e. Historical Data Services;
- f. External Data Services;
- g. Access Control Services;
- h. Manage Dispatch Services.



## 15. Database

The first application (vms-proxy-core-app) materialises the main functionality of VMS at EMSA proxy level, which is the management of VMS position reports (in NAF format) in a way that is independent from the channel of communication through which messages are exchanged.

The second application (ssn-si-core-app) materialises the main functionality of SSN-SI at national proxy level, which is the management of AIS position reports (in VDM format) (plus connection configuration settings, etc) in a way that is independent from the channel of communication through which messages are exchanged.

The third application (stires-core-app) materialises the main functionality of STIRES, which is the management of AIS messages (plus connection configuration/authentication settings, monitoring, etc) in a way that is independent from the channel of communication through which messages are exchanged.

The forth application (SSN GI) provides the graphical user interface (GUI) and handling user-to-business requests. It is the presentation layer providing the functionality of the VMS system.

The decomposition of VMS in three applications allows for:

- The disengagement of the business logic of VMS from the protocol for which it is offered. This disengagement, allows also for the independent implementation of the business logic of VMS.
- The development of the communication protocol has local repercussions in the corresponding application and not in the entire system (VMS).

The VMS system is identified by the following entities owned by the "VMS Management" Business System

- AIS Data/Position Report
- Ship Particulars/Identification (including/plus IR)
- Proxy Identification
- FMC Identification(Provider & Subscribers)
- "use" the ssn-domain/stires entities Vessel, Location, Proxy, AreasProxy
- VMS AIS Report Information that includes the above entities; i.e. it encapsulates the ships AIS position reports and the connections (FMCs Provider & Subscribers, Proxy, STIRES Core) information

Thus, SSN VMS system provides a web service that enables the access to, and update of, these entities as shown in Figure 4-18.

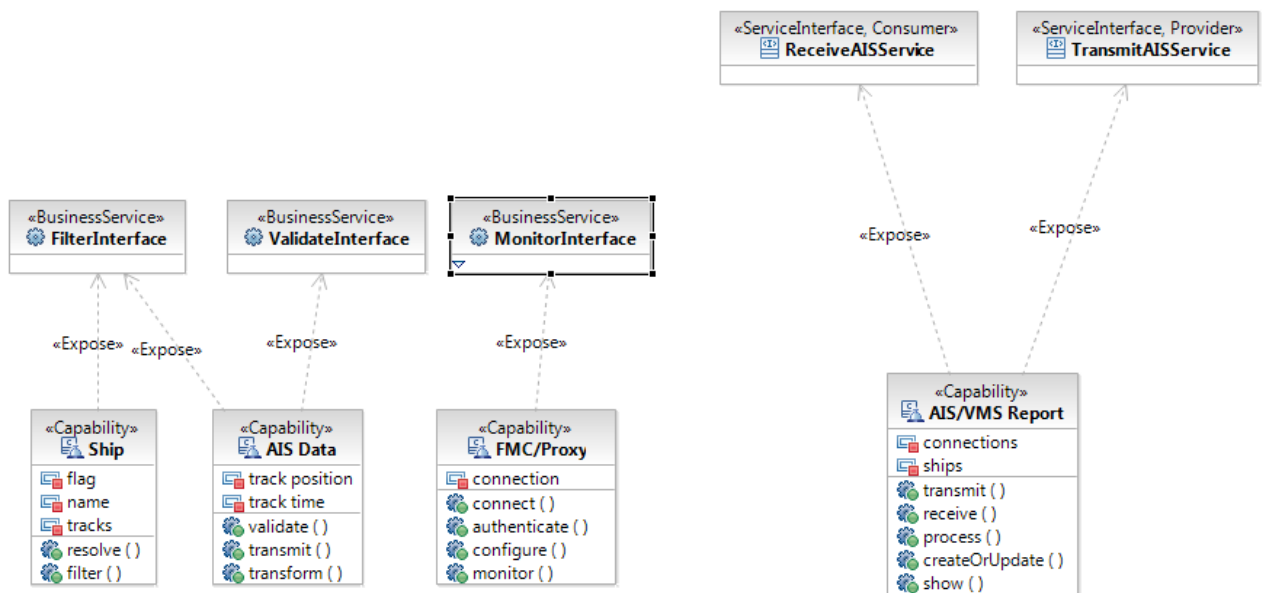


Figure 4-18SSN-SI VMS Service provided operations.

### 4.8.1 Domain module/package

This module/package includes the definition of VMS entities (position reports, vessel, proxy, etc) and it is used by all the VMS sub-systems.

An XSD (canonical format) shall be created based on the VMS domain named vms.xsd.

This schema shall be an enhanced version of ais.xsd that includes the AIS Position Reports and the additional information of FMCs connection properties (configuration data and statistics about data transmission).

The vms.xsd schema shall be based to ais.xsd so that the VMS canonical messages could be fast and easily transformed (XSLT) to LRIT and/or CSN corresponding ones.

#### 4.8.1.1 UML Class Diagrams

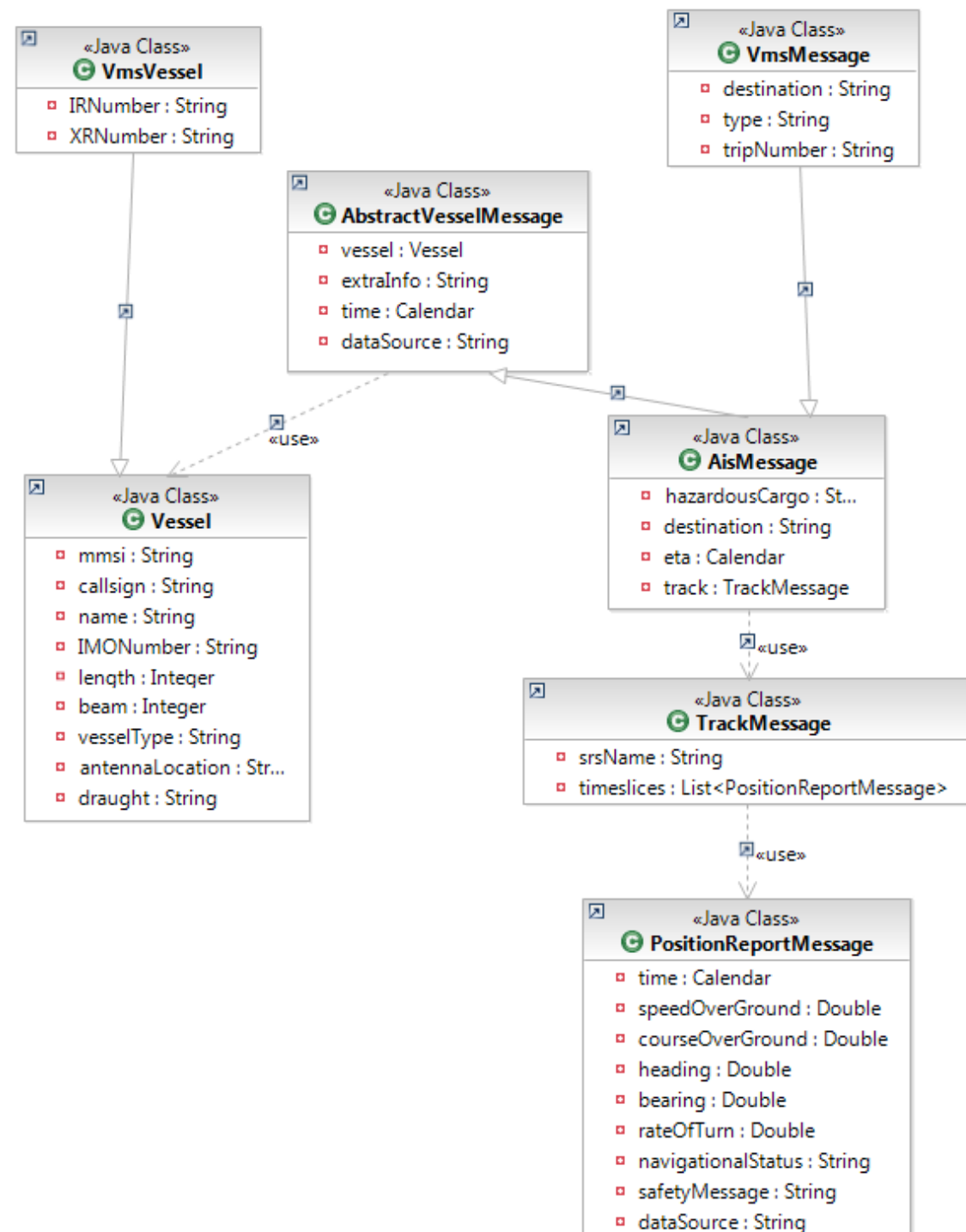
This section covers the architectural significant elements of the design model. It presents the definition of the most significant classes that will implement the requested functionality, organised into packages.

The classes are organised in packages according to the functionality they provide. A package is a general-purpose model element that organizes model elements into groups. Each package contains a set of classes and interfaces, representing what will become components in the implementation.

#### 4.8.1.2 Module: ssn-vms-domain

##### 4.8.1.2.1 Package: ssn.vms.domain

#### Class Diagram : vms vessel & message



#### Class

#### Vessel

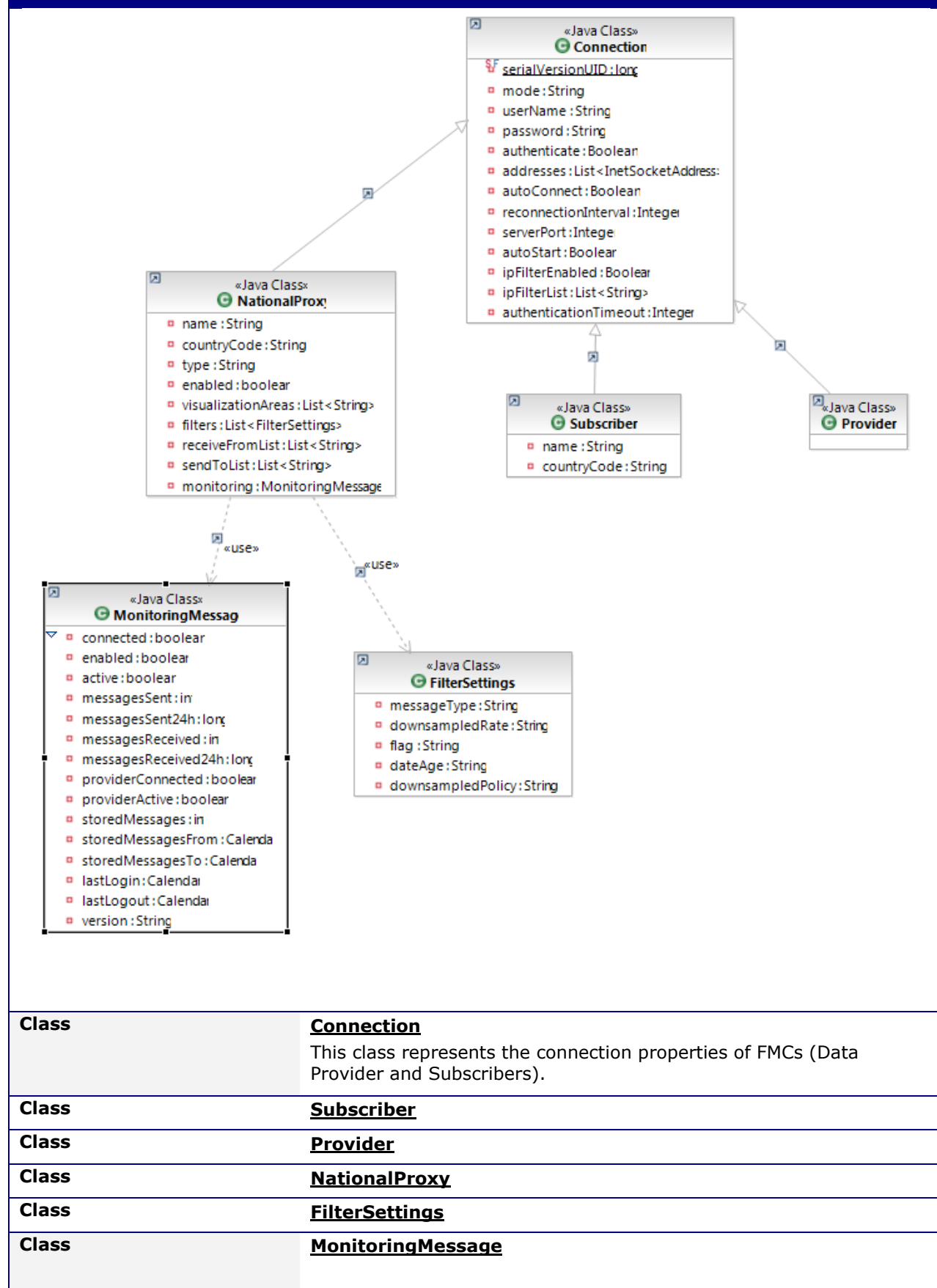
This class represents the vessel details.

#### Class Diagram : vms vessel & message

<b>Class</b>	<p><b><u>VmsVessel</u></b></p> <p>This class extends the aforementioned Vessel class to represent the additional VMS attributes such as</p> <ul style="list-style-type: none"> <li>➤ Internal reference No: Vessel registration detail. Unique vessel number as flag state Alpha-3 ISO country code followed by number.</li> <li>➤ External registration No: Vessel registration detail; the side number of the vessel.</li> </ul>
<b>Class</b>	<p><b><u>AbstractVesselMessage</u></b></p> <p>The abstract message related to a particular vessel. It includes the following attributes:</p> <ul style="list-style-type: none"> <li>➤ vessel: ship information</li> <li>➤ time: the message time stamp (in UTC timeframe)</li> <li>➤ dataSource: Organizational source of data for the message and may represent a data service provider, a data management system, an AIS transmission system, etc.</li> </ul>
<b>Class</b>	<p><b><u>AisMessage</u></b></p> <p>This class extends the aforementioned AbstractVesselMessage class to represent the additional AIS message attributes such as</p> <ul style="list-style-type: none"> <li>➤ eta: the expected time of arrival (in UTC timeframe)</li> <li>➤ track: the instance of TrackMessage class (refer below).</li> </ul>
<b>Class</b>	<p><b><u>TrackMessage</u></b></p> <p>This class is a sequence of specialized timeslices (PositionReportMessage) that indicate the dynamic status of the AIS message.</p>
<b>Class</b>	<p><b><u>PositionReportMessage</u></b></p> <p>This class encapsulates the various dynamic properties of AIS objects at a given point in time and space; it includes SOG, COG, ROT, longitude, latitude, true heading, data source.</p>
<b>Class</b>	<p><b><u>VmsMessage</u></b></p> <p>This class extends the aforementioned AisMessage class to represent the additional VMS message attributes such as:</p> <ul style="list-style-type: none"> <li>➤ destination: the address of the party receiving the message;</li> <li>➤ type: the VMS message type, e.g. 'POS';</li> <li>➤ tripNumber: Fishing trip serial number in current year.</li> </ul>

#### Class Diagram : vms connection & monitoring

## Class Diagram : vms connection & monitoring



## 4.8.2 Business Support module/package

This module/package provides/implements the common business processing as

- Down-sampling;
- Monitoring.

**Down-sampling** is performed by:

- data-age filter

A selective down-sampling will be performed on the AIS messages. For example, suppose that an AIS message with ITU number 5 is received from the AIS target 247234222 at time T0 and that this message is sent to the STIRES server. Later, at time T1, a new message with ITU number 5 is received from the same AIS target 247234222. Suppose that the parameter `downsampling_interval_itu_5` is set to 360 (seconds).

The possible situations are:

- |  |                               |
|--|-------------------------------|
| $T1 - T0 \leq \text{downsampling\_interval\_itu\_5}$ | The message will be discarded |
| $T1 - T0 > \text{downsampling\_interval\_itu\_5}$    | The message will be sent      |

- areas filter based on the policy filters.

## 4.8.3 VMS-Proxy

VMS-Proxy Architectural Layer Dependencies

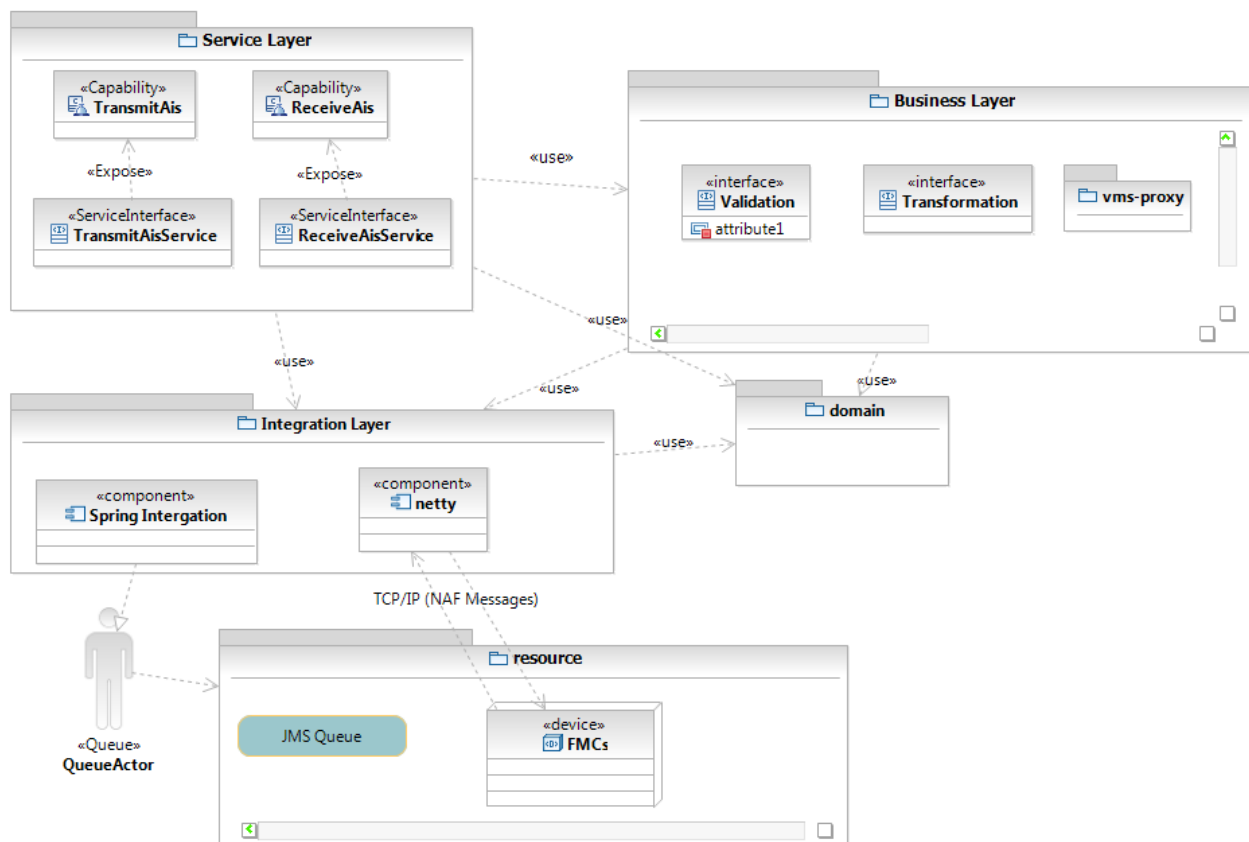


Figure 4-19 Architectural Layer dependencies (Logical view) of VMS-Proxy

The VMS-Proxy shall be implementing using Java and shall behave as **Light Service Bus** (providing features of Validation, Transformation, Queuing and Transportation using Spring Integration module).

#### 4. Integration Layer

##### TCP/IP connection

The Netty module is responsible to establish, manage and monitor the connection between the VMS-Proxy subsystem and the FMCs. It will implement a "keepalive" mechanism. It shall also provide the filtering of IP addresses if this feature is enabled for FMCs (Provider and Subscribers).

It shall establish TCP/IP (raw Socket) connection with the FMC application to receive (Data Provider) and send (Data Subscribers) NAF messages.

Two Buffers shall be used to store/buffer the incoming and the outgoing NAF messages.

##### Thread management

- The **incoming NAF messages** (from Data Provider stored to **Incoming NAF Buffer**) shall be processed by a pool of "consumers" (VMS-Proxy NAF Processor/Data Processing Service). Using the Spring Integration module; each NAF message shall be validated and transformed to AIS message (POJO) by the corresponding channels. Then, the AIS messages shall be packed (XML message) and sent to STIRES system via HTTPS.
- The **incoming AIS messages** (AIS messages from STIRES Server via HTTPS) shall be processed by a pool of "consumers" (VMS AIS Processor). Using the Spring Integration module; each AIS message shall be transformed (common transformer bean) to NAF message by the corresponding channels. Finally, the NAF messages shall be stored to the **Outgoing NAF Buffer**. A consumer shall dispatch the NAF messages to the FMCs application (Data Subscriber) - according to vessel flag and the coastal fishing zones determined by the STIRES Core - via TCP/IP (raw Socket) connection when the connection to FMC Server could be established.

#### 5. Business Layer

This layer provides the Data processing such as the message validation, transformation.

#### 6. Service Layer

This layer exposes the VMS-Proxy functionality as RESTful Services.

**Transmit AIS Report capability;** it sends the AIS Report messages to STIRES Core System – XML messages over HTTPS.

The AIS Report messages include the Position Reports received from Data Providers.

**Receive AIS Report capability;** it accepts the AIS Report messages from STIRES Core System – XML messages over HTTPS.

The AIS Report messages includes the Position Reports that STIRES received from sources different than FMCs.

### 4.9 SSN-Blue Belt

This section provides an overview of the SSN Blue Belt System. The current SSN system (EIS and STIRES) is lightly upgraded (an application parameter shall be used to enable/disable the Blue Belt behaviour) to incorporate the required Blue Belt changes. The SSN Blue Belt System shall use the current SSN GIS Services to provide the related functionality.

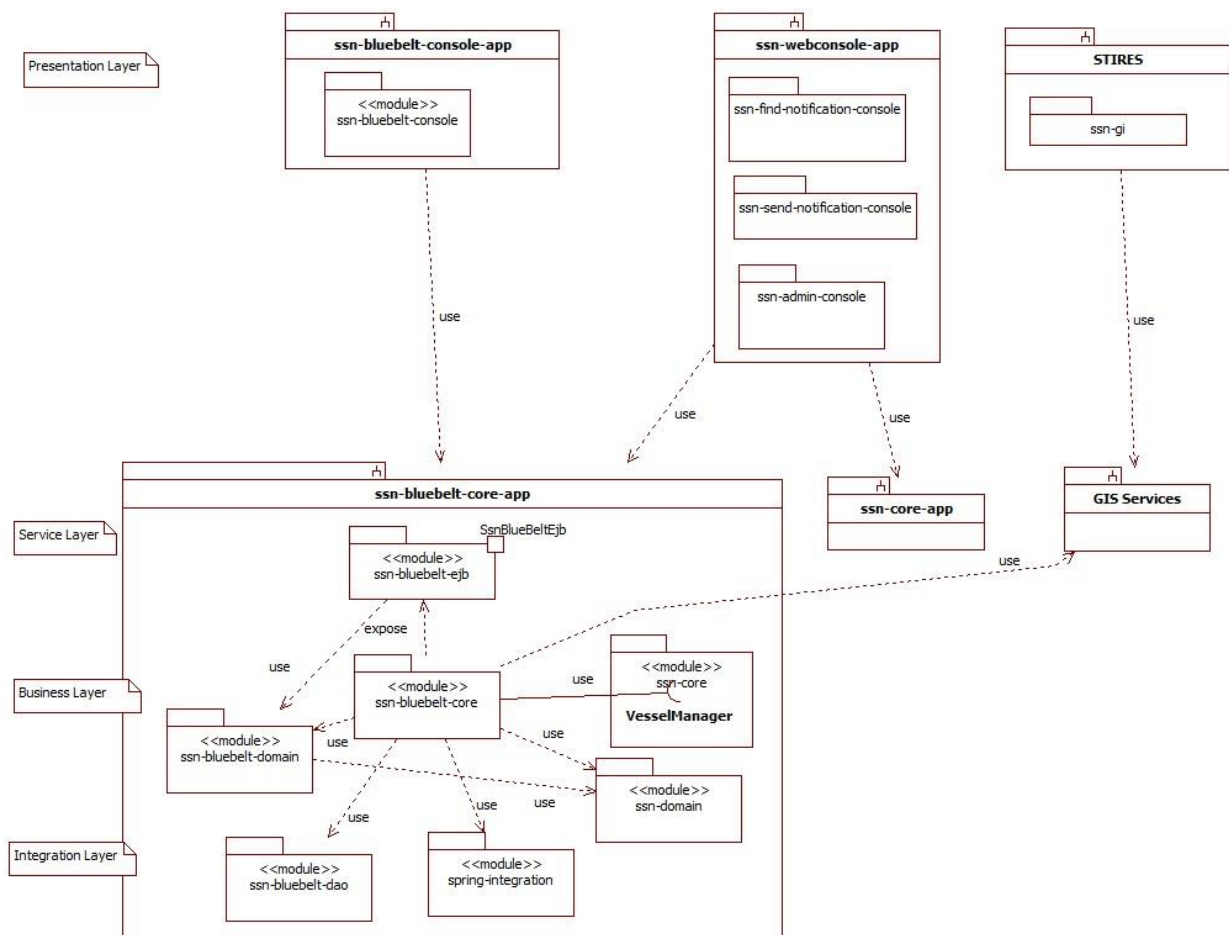
A sovereign element in the design of the Blue BeltSystem (hereinafter BB), as depicted in the logical view diagram presented in Figure 4-20 is the division of system in two distinguishable applications:

- **ssn-bluebelt-core-app** (ref:section 4.2)
- **ssn-bluebelt-console-app** (ref:section 4.3)

The **ssn-console-app** shall be **upgraded** to provide the BB functionality. The following modules shall be upgraded

- ssn-admin-console: Web application (SSN Administrator Web Application), which will be used for the management of SSN resources such as the shipping companies, Blue Belt ships and parties as well as access rights concerning the Blue Belt pilot/customs authorities.
- ssn-send-notification-console: Web application, which will be used to send customs consignment notifications
- ssn-find-notification-console: Web application, which will be used to find the relative ship calls of Blue Belt ships.

The **SSN GI** shall be also **upgraded** to provide the visualization of Blue Belt functionality.



**Figure 4-20 Architectural Layer dependencies (Logical view)**

The first application (ssn-bluebelt-core-app) materialises the main functionality of BB, which is the management of BB entities (Vessel, Customs Information, Shipping Company) in a way that is independent from the channel of communication through which messages are exchanged.



The second application (ssn-bluebelt-console-app) provides the graphical user interface (GUI) and handling user-to-business requests. It is the presentation layer providing the functionality of the BB system.

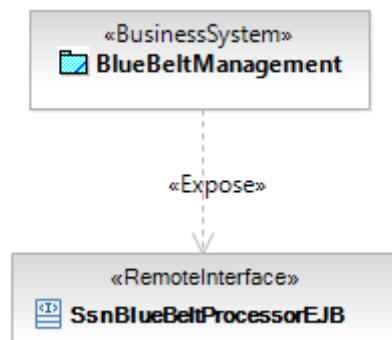
The decomposition of BB in two applications allows for:

- The disengagement of the business logic of BB from the protocol for which it is offered. This disengagement, allows also for the independent implementation of the business logic of BB.
- The development of the communication protocol has local repercussions in the corresponding application and not in the entire system (BB).

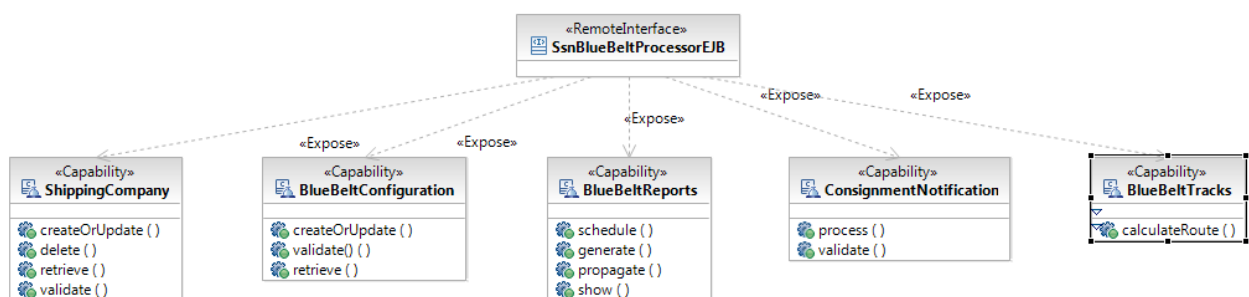
The BB system is identified by the following entities owned by the "BlueBelt Management" Business System

- "Blue Belt Vessel Configuration" (blue ship monitoring period)
- "Shipping Company"
- "Customs Information"
- "use" the ssn-domain entities Vessel, Voyage, Location, Parties
- Blue Belt Report Information that includes the above entities

Thus, SSN BB system provides a service (EJB 3.0) that enables the access to, and update of, these entities as shown in the Figure 4-2 and Figure 4-22.



**Figure 4-21 SSN BB Services**



**Figure 4-22 SSN BB Service provided operations.**

#### 4.9.1 SSN Blue BeltCore Application - ssn-bluebelt-core-app

## `\${ssn-bluebelt-core-app}` Package Dependencies

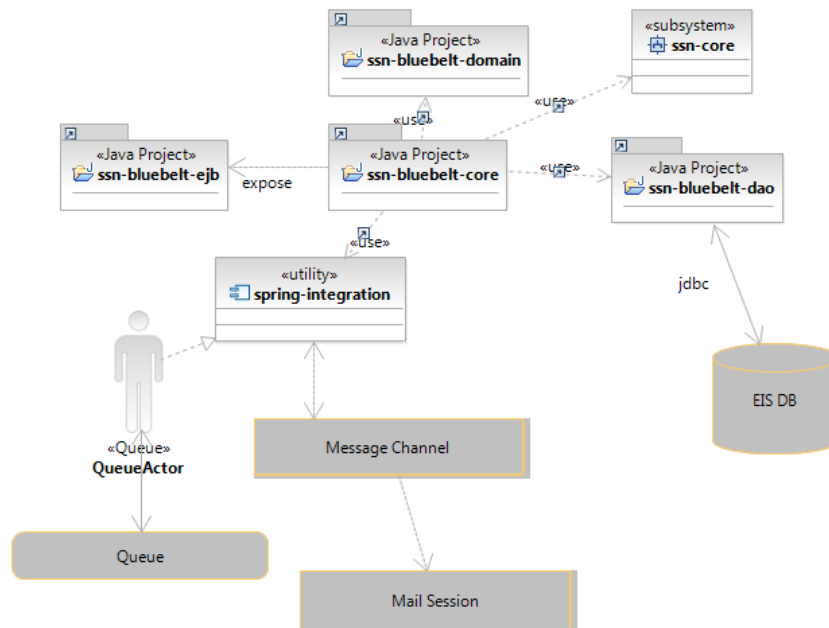


Figure 4-23 Package Dependencies

The application is constituted from the modules:

7. **ssn-bluebelt-domain**: SSN BB Business Domain Objects Modules. It includes the SSN BB Domain objects designed using Plain Java Classes and Interfaces. The SSN BB Domain objects encapsulate the state and behaviour of business entities. Examples of business entities in a SSNBB application are BlueBeltConfiguration, ShippingCompany, ConsignmentNotification, etc. This module is used by all the other modules of SSN-BB system.
8. **ssn-bluebelt-dao**: This module implements the EIS data access. It *decouples application code from data access code*.
9. **ssn-bluebelt-core**: It is the heart of application. This module implements the SSNBB functionality.
10. **ssn-bluebelt-ejb**: It is a lightweight EJB that lends Remote Access semantics to the main services/operations of ssn-bluebelt-core. It exposes the remote stateless session EJB **SsnBlueBeltProcessorBean**.
11. **spring-integration**: This library provides support for management of JMS queues, the submission of the e-mails.

### 4.9.1.1 UML Class Diagrams

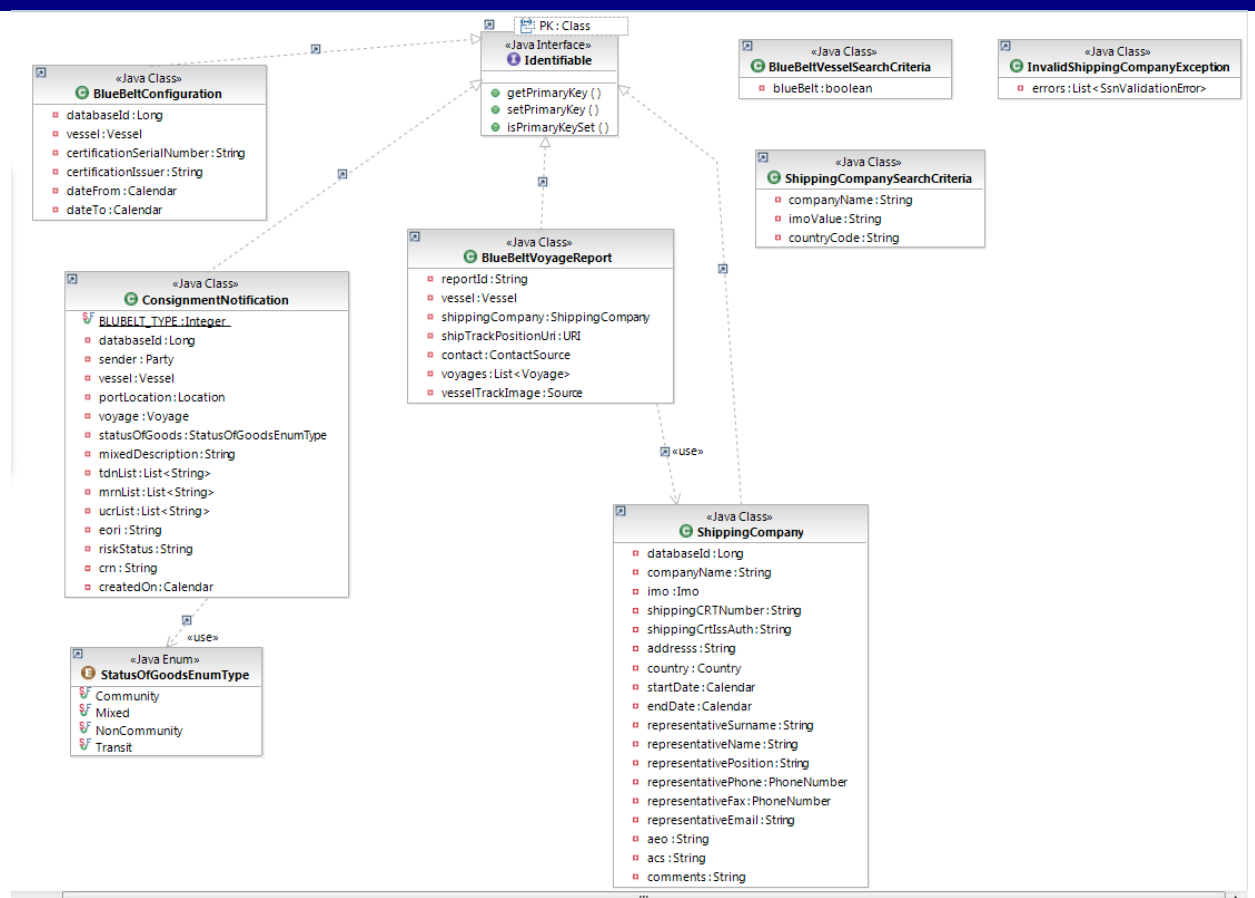
This section covers the architectural significant elements of the design model. It presents the definition of the most significant classes that will implement the requested functionality, organised into packages.

The classes are organised in packages according to the functionality they provide. A package is a general-purpose model element that organizes model elements into groups. Each package contains a set of classes and interfaces, representing what will become components in the implementation.

#### 4.9.1.2 Module: ssn-bluebelt-domain

##### 4.9.1.2.1 Package: bluebelt

**Class Diagram : bluebelt**

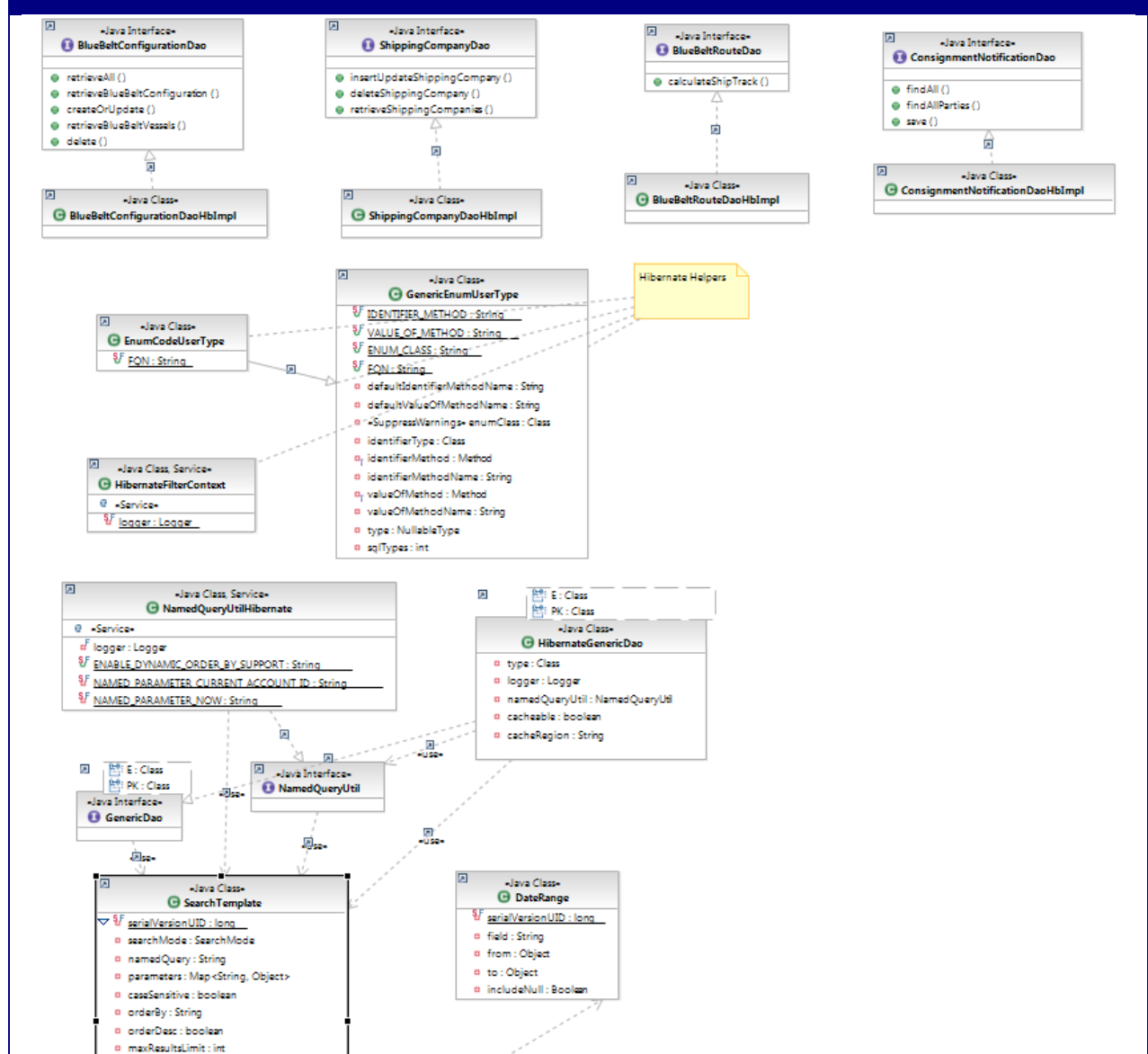


<b>Class</b>	<b><u>ShippingCompany</u></b> This class represents the shipping company details.
<b>Class</b>	<b><u>BlueBeltConfiguration</u></b> This class represents the BlueBelt Configuration details.
<b>Class</b>	<b><u>ConsignmentNotification</u></b> This class represents the Consignment data notification.
<b>Class</b>	<b><u>BlueBeltVoyageReport</u></b> This class represents the Blue ship voyage report content.
<b>Class</b>	<b><u>ShippingCompanySearchCriteria</u></b> This class represents the Shipping Company Search Criteria.
<b>Class</b>	<b><u>BlueBeltVesselSearchCriteria</u></b> This class represents the Blue Belt Vessel Search Criteria taht extends the EIS VesselSearchCriteria.
<b>Class</b>	<b><u>InvalidShippingCompanyException</u></b> A business Exception class used by validators.
<b>Interface</b>	<b><u>Identifiable</u></b> A helper interface related to the identifier property of the entity.

#### 4.9.1.3 Module ssn-bluebelt-dao

##### 4.9.1.3.1 Package: bluebelt-dao

##### Class Diagram : bluebelt-dao



<b>Interface</b>	<b>BlueBeltConfigurationDao</b> A data access object for managing the database operations for Blue ships
<b>Class</b>	<b>BlueBeltConfigurationDaoHbImpl</b> Hibernate based implementation of the BlueBeltConfigurationDao.
<b>Interface</b>	<b>ShippingCompanyDao</b> A data access object for managing the database operations for ShippingCompanies.
<b>Class</b>	<b>ShippingCompanyDaoHbImpl</b> Hibernate based implementation of the ShippingCompanyDao.
<b>Interface</b>	<b>ConsignmentNotificationDao</b> A data access object for managing the database operations for customs information.

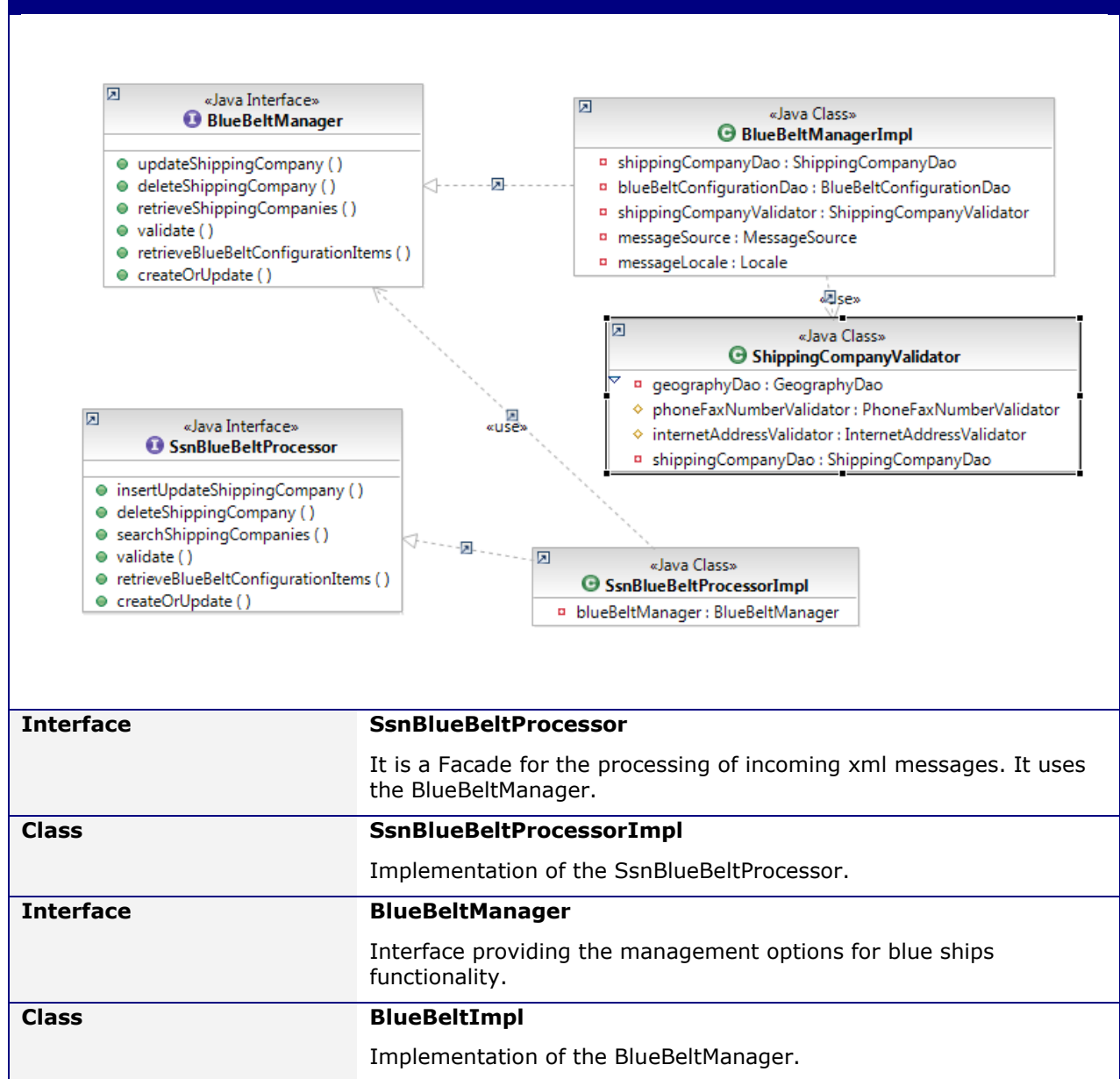
#### Class Diagram : bluebelt-dao

<b>Class</b>	<b>ConsignmentNotificationDaoHbImpl</b> Hibernate based implementation of the ConsignmentNotificationDao.
<b>Interface</b>	<b>BlueBeltRouteDao</b> A data access object for calculating the blue ship tracks based on STIRES spatial data.
<b>Class</b>	<b>BlueBeltRouteDaoHbImpl</b> Hibernate based implementation of the BlueBeltRouteDao.

#### 4.9.1.4 Module: ssn-bluebelt-core

##### 4.9.1.4.1 Package: bluebelt-manager

#### Class Diagram : bluebelt-manager



#### 4.9.1.4.2 Package: bluebelt-validation

Class Diagram : bluebelt-validation	
<b>Class</b>	<b>ShippingCompanyValidator</b>
	Implements the ShippingCompanies business rules.

## 4.9.2 BB Console Application - ssn-bluebelt-console-app

The application is constituted from the **ssn-bluebelt-console**. It is a Web GIS application, which will be used by the SSN users for the Plotting of the route of blue ships. It will be implemented based on the SSN GI technology and it will provide a subset of the functionality of the current version of SSN GI. It will use the GIS Services (the C-MAP WMS available at EMSA) in the same way as the current version of SSN GI.

The BB web application does not contain distinguishable components and is simply a UI. It is simply using the functionality provided by the ssn-bluebelt-core-app.

### 4.9.2.1 Security on the ssn- bluebelt -console-app

The ssn-core-app application security relies on:

- The system level security provided by the runtime environment (SSL and Client Certificate).
- The users' authentication provided by the SSN SSO.
- The users' authorization provided by ssn-core module of the current EIS System.

### 4.9.2.2 UML Interfaces between the ssn-bluebelt-console-app and the ssn-bluebelt-core-app

In general, the **ssn-bluebelt-core-app** application, accepts calls for processing synchronous messages via the remote stateless session EJB **SsnBlueBeltProcessorBean**, which actually is a proxy of the class **SsnBlueBeltProcessorImpl**.

## 4.9.3 Message Queue

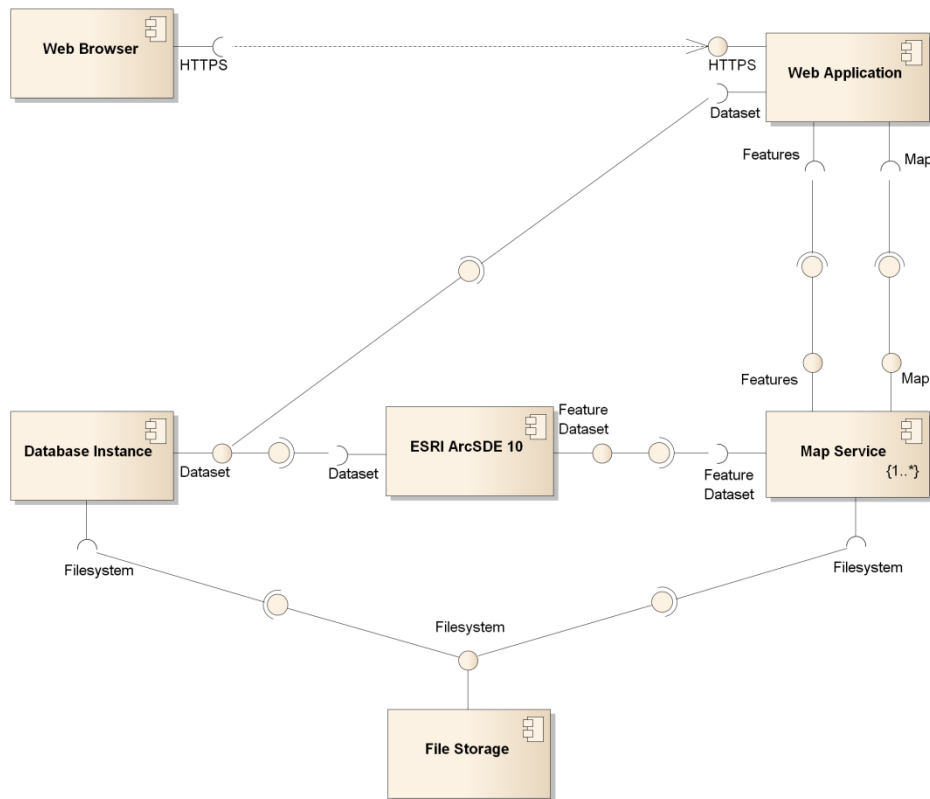
The ssn-bluebelt-core-app application uses a persistent distributed queue – namely “BB warning messages” – in order to communicate in an asynchronous manner with the channel propagate the warning messages to the corresponding recipients. This queue offer, basically, point-to-point<sup>4</sup> communication. The queue messages payload is of type *BlueBeltVoyageReport*.

## 4.10 SSN GI

The main aim of this intervention is to implement a new visualisation mechanism that will allow for a more efficient management and graphical rendering of spatial information and, in particular, of the large amount of vessel position data that are constantly being entered into the system. The goal of the new design is not to dismiss or provide replacements for all aspects of the existing system's architecture. Rather, it intends to improve on the current architecture by adopting previous design decisions and elements that were based on sound software development principles

<sup>4</sup> A Point-to-Point Channel ensures that only one receiver consumes any given message.

and technologies, and modify only those aspects of the system that it is required in order to meet the new specifications. As such, viewing the new system from the perspective of its conceptual architecture, as displayed in Figure 4-24, the new design is consistent with that of the existing system. Deviations to the original concept become apparent at a lower level and are limited to the visualisation scheme utilised for rendering spatial information, while other design aspects relating to the business domain and data access layers remain mostly the same.



**Figure 4-24 Component diagram providing a high-level view of both the existing and the new system's architecture**

The following section presents the design of the new visualisation scheme for spatial information, which includes the implementation of a client-based, graphic rendering engine, as part of a new, redesigned user interface, the consolidation and reorganisation of the existing map services and layers deployed via ArcGIS Server and the remodelling of the related parts of the existing system's database.

#### 4.10.1 Spatial Information Visualisation Scheme

As previously mentioned, the new system intends to replace the existing system's mechanism for visualising spatial information through the introduction of an improved graphic rendering engine. The need for a new rendering engine stems from the fact that, in its current form, the application depends on ArcGIS server's own drawing engine for rendering both static (e.g. world map, maritime information, ports etc.) and dynamic (e.g. vessel tracks, routes etc.) geographic information. Even though this technique works sufficiently well for displaying mostly static information with low user interactivity requirements, when it comes to displaying dynamic information via a feature-rich GUI it becomes largely inefficient. This is due to the fact that, since the visual information is prepared and rendered on the server side, the client (web-browser) is unaware of the kind of information that is being displayed. Consequently, even for the simplest of user actions like hovering over a vessel track, the browser resorts to having to perform multiple queries to the server in order to discover what information is contained on the application's viewport.

In order to enable the application to provide users with a more feature-rich GUI with a high rate of interactivity and low latency between user actions, it is inevitable that a lot of the information rendering functionality is transferred from the server to the web-browser. In this way, the latter becomes more aware of the information it is required to display and can minimise the amount of round-trips to the application server as a result of the queries that need to be performed, improving the general performance of the application by orders of magnitude. Also, since the required geometrical and textual attributes of the vessel tracks will be transferring to the web-browser for rendering, many of the ArcGIS server mapping services/layers that are currently used for distinguishing between the various types of data sources and vessel types and for indicating enrichment attributes and labelling options can be discarded as they will no longer be necessary. As a result, the remaining services will be reorganised so that they provide much more efficient access to the database, while the latter's schema will be remodelled and become normalised, so as to avoid having to perform multiple database queries in order to retrieve information from multiple data sources, as in the existing system. The proposed modifications will also aid in increasing application performance by lessening the burden of the application, GIS and database servers.

#### **4.10.2 Structure**

In the new design, as shown in Figure 4-25, even though the map-viewer control is still directly responsible for requesting and displaying background, base-map images from the respective web services, requesting image overlays that contain vessel and port icons are no longer part of its duties. This responsibility now lies with the new, JavaScript-based rendering engine, whose major task is to perform the appropriate queries requesting spatial feature data instead of tiled image overlays. This data is consequently processed and the appropriate vessel and port icons are created which are then inserted in the graphics layer of the map-viewer control as interactive graphical objects.

Displaying graphical and textual decorations over vessel and port icons is also part of the functionality of the client rendering engine, a fact that is reflected by the significantly smaller number of layers that are required in the new visualisation scheme. Another important aspect of the new scheme is the fact that, concerning the vessel data sources, i.e. AIS, LRIT and VMS, there is no longer a distinction between them at the map service level and also, partly, at the database level. In the new system, the vessel data source becomes just another attribute of all vessel tracks which are maintained in a collective fashion. In comparison with the current system, the proposed consolidation of map services and database tables will result in a threefold decrease of the total number of vessel related web and database queries that are required to fully render a map instance at runtime.



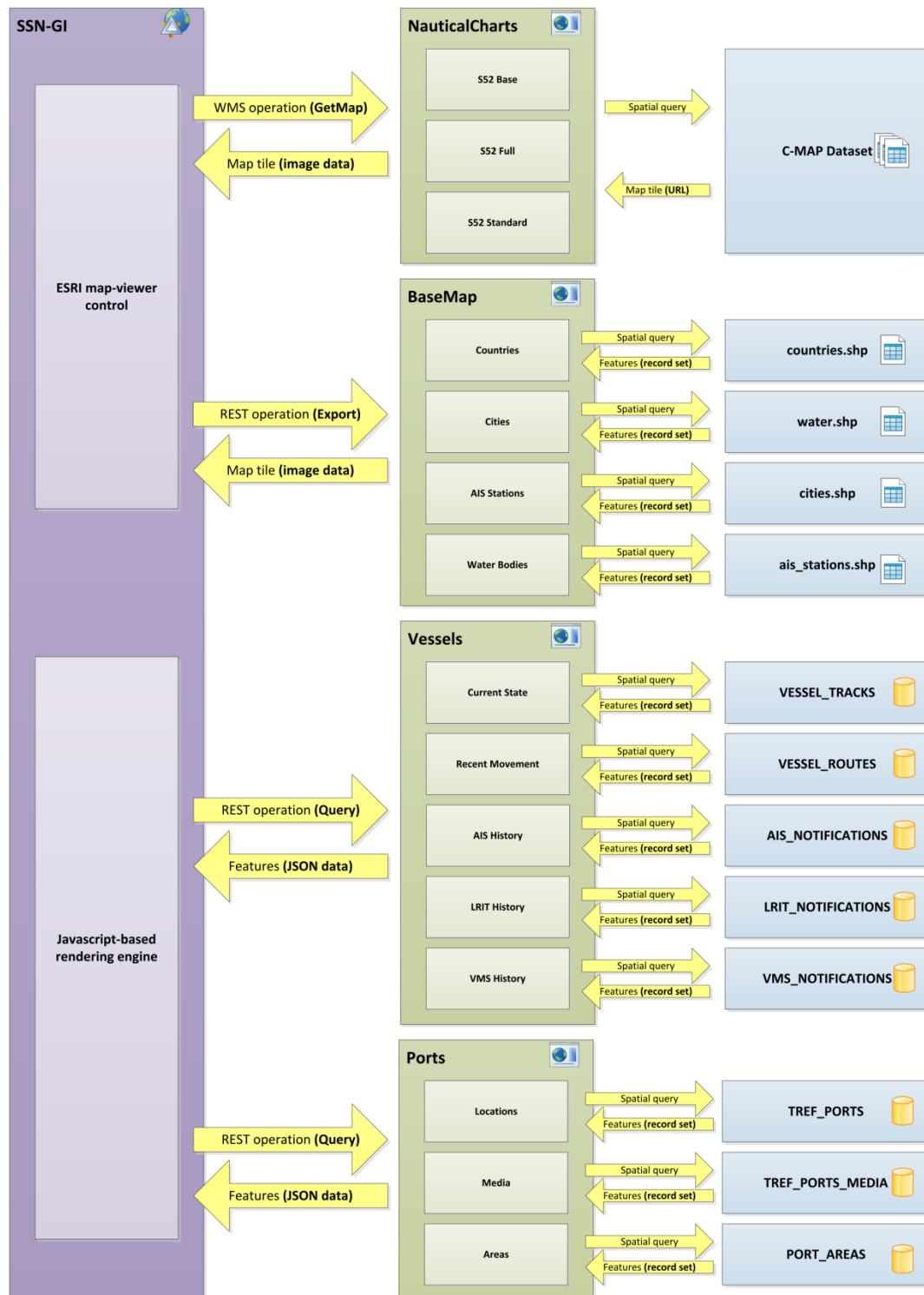


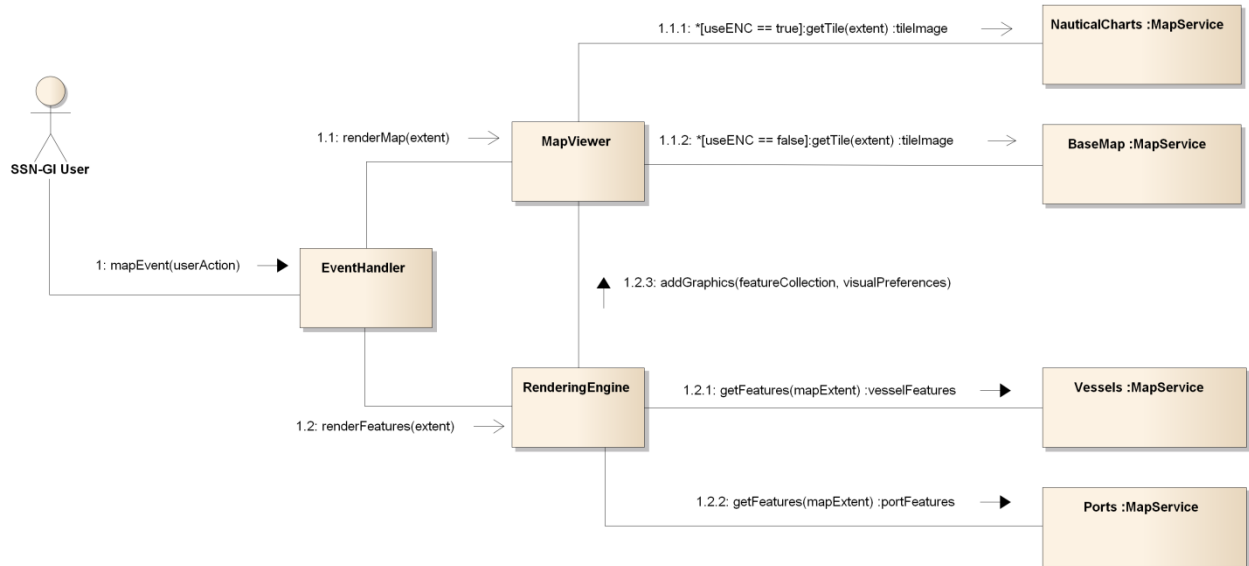
Figure 4-25 Interaction paths between the various elements of the new system's spatial information visualisation scheme

### 4.10.3 Behaviour

The graphical rendering of spatial information in the context of the new visualisation scheme is split into two distinct phases:

- rendering geographical features as static information, placed in the background
- rendering vessel features, port features and other graphical shapes as interactive objects, in the foreground

As shown in Figure 4-26, both of these processes execute asynchronously, triggered by user actions such as panning or zooming that modify the map's viewable area, i.e. the map's extent.



**Figure 4-26 Communication diagram of a typical map rendering cycle**

For the first phase, the same mechanism that is currently in use in the existing system is also employed in the new one, i.e. user actions changing the map's extent like panning and zooming, become delegated to the map-viewer web control, which in turn depends on appropriate HTTP queries to web services for retrieving tile images. Since the information to be displayed during this phase is static by nature, map rendering performance is generally dependent on I/O bandwidth since most or, in some cases, all images returned to the map-viewer control by the web services are retrieved from spatially aware file caches containing pre-rendered tiles. There are various factors affecting the population scheme of these caches, i.e. whether the entire dataset is pre-rendered in advance or whether it is generated on demand, however, since they do not have direct impact on the application's design, these factors will all be considered during the project's implementation phase and an appropriate configuration will be put in place.

For the second phase, the custom designed rendering engine performs spatial queries to the web mapping services that, instead of returning rendered image overlays for the map, they rather return feature objects serialised in JSON format. These objects undergo further processing in order to be converted into graphical objects, suitable for being overlaid on the map and exhibiting interactive functional attributes such as context menus, custom cursors, pop-up windows etc. Since the generated graphical objects are inserted in the DOM tree of the client browser, in order not to hinder the performance of the latter, a special clustering algorithm will be employed by the rendering engine for cases where the number of objects exceeds a pre-specified limit. This algorithm will be applied specifically on vessel track icons, which in particular zoom levels and/or port areas can amount to a range of many thousands. In those cases, neighbouring icons are clustered into single, larger graphical objects that have the form of rectangles, labelled using the total number of clustered tracks.

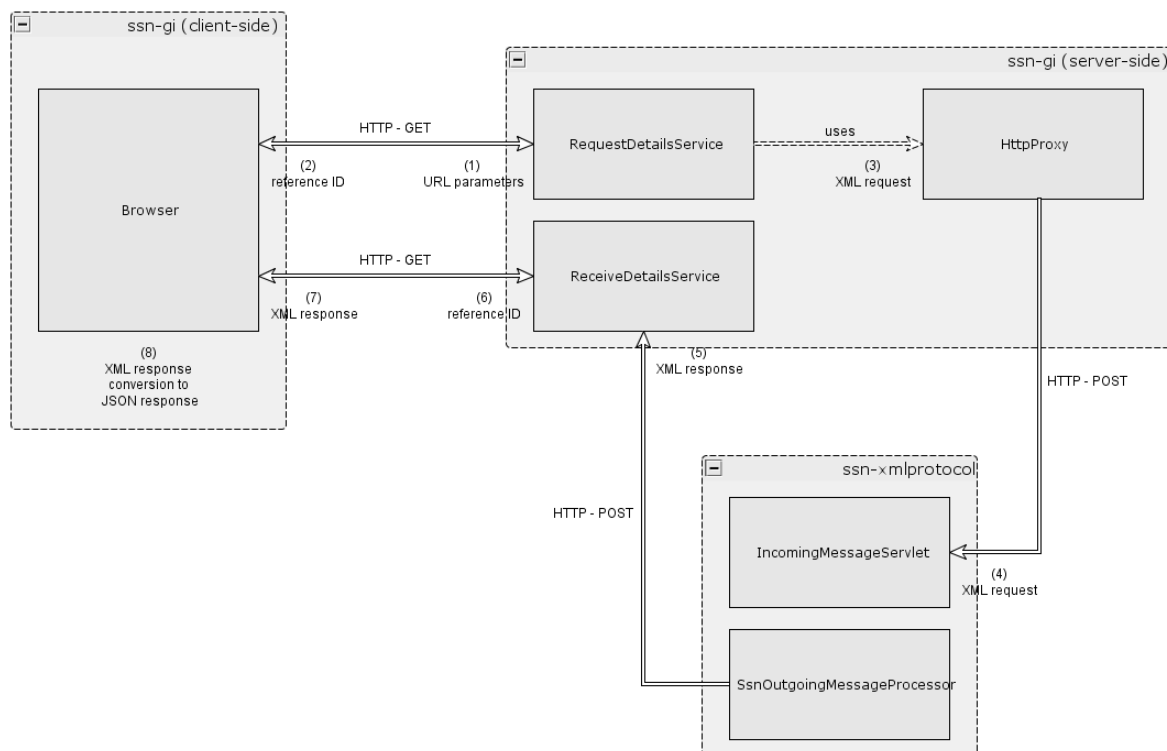
The mechanism for rendering other user drawn graphical shapes such as points, lines, polygons and text, as well as newly introduced object types such as placemark icons, remains the same as in the existing application. Those objects stored to and retrieved from the database directly in the form of JSON formatted strings, without relying on map services.

## 4.11 SSN-GI / EIS notification details request protocol mechanism upgrade

This section describes the mechanism of communication between the SSN-GI and EIS applications and, in particular, the request/response protocol of EIS notification details exchange. The current situation is presented, identifying the various shortcomings of the present scheme and the proposed improved solution in order to overcome them.

### 4.11.1 Current mechanism

In its current form, the mechanism by which the SSN-GI application requests notification details from EIS is realised through the asynchronous exchange of XML protocol request/response messages (Figure 4-27). A notification details request is initiated by the SSN-GI front-end (web browser) as a parameterised HTTP GET request. Once received by the SSN-GI back-end, the request parameters become marshalled into an XML request message that is in turn sent as payload of an appropriate HTTP POST request to the EIS back-end for further processing.



**Figure 4-27 Current EIS notification request protocol mechanism (numbers in parentheses are indicative of the sequence of information delivery – operations are otherwise asynchronous)**

After the message has been processed in EIS, a new HTTP POST request is initiated towards the SSN-GI back-end, whose payload consists of the notification details XML response message. Because of the asynchronous nature of this mechanism, during the time that a request is being processed by EIS, the browser periodically polls the SSN-GI server using a reference ID number that was assigned to the original request, until the appropriate response message from EIS becomes available, in which case it is passed as the response to the browser. There it becomes transformed into a JSON response object to be used for populating the appropriate GUI elements.

Even though this mechanism is computationally less expensive than the respective mechanism of previous versions of WEB-GI v2.1, which also involved additional steps in message transport

through JMS queues, it is still not the most efficient way to transfer notification details information between GI and EIS. The participation of the *ssn-xmlprotocol* module in the current mechanism introduces many unnecessary conversion and mapping operations between URL parameters, application domain objects, protocol objects (JAXB), XML content and JSON. Also, the asynchronous processing implemented by the *ssn-xmlprotocol* enforces the use of polling from the client-side that, under certain operational conditions, can potentially hinder application performance.

#### 4.11.2 Proposed improvement

In the solution proposed herein, it is suggested that the *ssn-xmlprotocol* module becomes excluded from the new mechanism and the whole process becomes replaced by a synchronous HTTP transaction between the SSN-GI front-end and EIS itself, with the SSN-GI server assuming the role of an intermediary component between the two (Figure 4-28). In the new scheme, the HTTP GET notification details request originating from the browser is sent as-is to EIS by the SSN-GI back-end, which acts as a simple, protocol-agnostic proxy. This solution removes SSN-GI back-end dependencies to JAXB generated objects, used for marshalling requests into XML messages and also introduces the *ssn-message-web* EIS module. It acts as an HTTP “bridge” between various other system components and the EIS application domain, providing information in browser-friendly JSON format. It shall be noted that this mechanism can be utilized for the interface between SSN-GI and EIS to exchange all types of SSN request & response messages.

The proposal ensures that the 2 systems (SSN-GI and SSN-EIS) can be deployed on different servers.

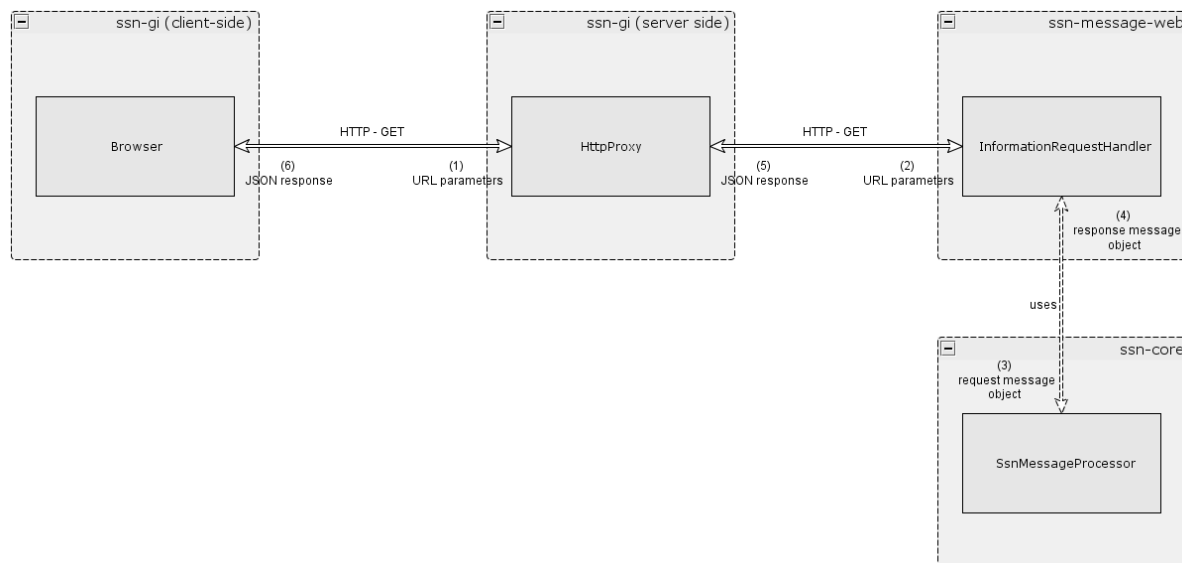


Figure 4-28 Proposed EIS notification request synchronous protocol mechanism

#### 4.12 Message Queues

The *ssn-core-app* application uses three queues – namely incoming, outgoing and notification propagation messages queue – in order to communicate in an asynchronous manner with its clients (in this case the *ssn-xmlprotocol-app*). These queues offer, basically, point-to-point<sup>5</sup> communication.

<sup>5</sup> A Point-to-Point Channel ensures that only one receiver consumes any given message.

Each queue is capable of accepting multiple types of messages:

Queue	Accepted Message Types
Ingoing Queue	<ul style="list-style-type: none"> <li>• <b>InformationRequest</b> (MS2SSN_&lt;SSN_type&gt;_Req)</li> <li>• <b>AdditionalInformationReply</b> (MS2SSN_&lt;SSN_type&gt;_Res)</li> </ul>
Outgoing Queue	<ul style="list-style-type: none"> <li>• <b>AdditionalInformationRequest</b> (SSN2MS_&lt;SSN_type&gt;_Req)</li> <li>• <b>InformationReply</b> (SSN2MS_&lt;SSN_type&gt;_Res)</li> </ul>
Notification Propagation Queue	<ul style="list-style-type: none"> <li>• <b>Notifications</b> <ul style="list-style-type: none"> <li>○ Vessel notifications with sat service</li> <li>○ Alert notification</li> <li>○ Alert Distribution notification</li> </ul> </li> </ul>

**Table 4-10 Types of messages per Queue**

Additionally, the ssn-core-app application uses a forth queue – named WaitForReply queue – in order to keep a copy of the **AdditionalInformationRequests** that performs to a data provider. Periodically (refer also to 4.2.1.5), ssn-core-app polls this queue in order to discover AdditionalInformationRequest messages for which the responsible data provider MS has not reply within a time interval. This time interval is the timeout value of the corresponding request message. It should be noted that this timeout value is also included in the generated request message for additional information.

Furthermore, ssn-core-app when processing a data provider reply (AdditionalInformationReply), uses the WaitForReply queue to find the initial data request – stored within a AdditionalInformationRequest – in order to produce the appropriate InformationReply message.

In other words the wait for reply queue is mainly a temporary storage. The major advantage of adopting a queue as a temporary storage as opposed to the use of the RDBMS is that the application server effectively serializes (using java serialization) the messages and there is no need to develop and manage a separate a persistence layer.

End points:

- The ssn-core-app application maintains a permanent listener that consumes messages from the incoming message queue and following the "[process manager](#)" design pattern. The listener is implemented as the *Message Driven Bean* **IncomingMessageProcessorMdb** located under the package **ssn.message.processor.ejb**, which is a proxy of the actual process manager **IncomingRequestReplyEnvelopProcessor** located at the package **ssn.message.processor**. It should be noted that in case of any exception during the processing of the message, this message is discarded.
- This process manager uses the following routing table:

Message Type	Processor
<b>InformationRequest</b> (MS2SSN_<SSN_type>_Req)	<b>ssn.request.processor.RequestProcessorImpl</b>
<b>AdditionalInformationReply</b> (MS2SSN_<SSN_type>_Res)	<b>ssn.reply.processor.ReplyProcessorImpl</b>

**Table 4-11 Process manager – routing table**

## 4.13 LOCODE Management – Upload LOCODEs

Business process specifics	
Responsibilities	<ul style="list-style-type: none"> <li>The responsibility is twofold: <ol style="list-style-type: none"> <li>Load onto the database a list of Location Codes that originate from the site of UNECE.</li> <li>Load onto the database a list of Location Names.</li> </ol> </li> <li>The list of codes is provided in a predefined comma separated (CSV) file format while the list of names could be provided in TXT or CSV file format.</li> </ul>
User role(s)	The EIS system administrator performs the task upon request.
Action History	<ul style="list-style-type: none"> <li>The action will be logged into the database log tables (TLOG).</li> <li>A log file will be created listing the processing of each record.</li> </ul>
Processing logic	<ul style="list-style-type: none"> <li>Two Oracle database External tables exist that map the definition of the CSV file for the UNECE Location and the Location Names. External tables make use of the SQL Loader functionality.</li> <li>The uploaded CSV file is stored as blob on SSN database. The "upload" action triggers the procedure being described.</li> <li>The uploaded file is extracted as ASCII file on an Oracle directory and the external table is altered so to access the data of the extracted file.</li> <li>The data of the external table are copied to the stage area. On copy, a flag is initiated that indicates when a location code already exists on the registry. The EIS system administrator validates the data on stage area and classifies the location codes to be stored on Locations registry or to be ignored.</li> <li>After the EIS system administrator validation, the data from the stage area are merged to the operational tables. New location codes are inserted. Existing location codes are updated: LOCATION_NAME, LONGITUDE, LATITUDE.</li> </ul>
Validation Rules	<ul style="list-style-type: none"> <li>Location codes to be loaded must be Port specific (Port = 1).</li> <li>The origin UN of each record will be recorded in the database.</li> <li>All the new LOCODEs will be activated upon administrator validation.</li> <li>Longitude and Latitude values must be converted to the SSN supported format.</li> </ul>
Database Transactions	<ul style="list-style-type: none"> <li>Create an Oracle directory that points to an existing path in the hosting server. The Oracle database owner must have full access to that path.</li> <li>Create an External table to map the definition of the CSV file.</li> <li>Create a stage area to allow the EIS system administrator identifies the Location codes of the CSV file to be stored on Locations registry.</li> <li>Merge data in the operation tables that store the definition of Location codes.</li> </ul>

## 4.14 Vessel Management – Upload Single Hull Tankers

Business process specifics	
Responsibilities	<ul style="list-style-type: none"> <li>Load onto the database a list of Single Hull Tanker.</li> <li>The list of codes is provided in a predefined comma separated (CSV) file format.</li> </ul>
User role(s)	The EIS system administrator performs the task upon request.
Action History	<ul style="list-style-type: none"> <li>The action will be logged into the database log tables (TLOG).</li> <li>A log file will be created listing the processing of each record.</li> </ul>
Processing logic	<ul style="list-style-type: none"> <li>An Oracle database External table exists that maps the definition of the CSV file. External table makes use of the SQL Loader functionality.</li> <li>The uploaded CSV file is stored as blob on SSN database. The "upload" action triggers the procedure being described.</li> <li>The uploaded file is extracted as ASCII file on an Oracle directory and the external table is altered so to access the data of the extracted file.</li> <li>The data of the external table are copied to the stage area. On copy, a flag is initiated that indicates when a vessel already exists on the registry. The EIS system administrator validates the data on stage area and classify the vessels to be stored on Vessels registry or to be ignored.</li> <li>After the EIS system administrator validation, the data from the stage area are merged to the operational tables. New vessels identified by IMO numbers that do not exist on the Vessels registry, are inserted; one record is created on the Vessels table including the IMO number and a second one on the Vessel_Detail_Versions table including the MMSI number, the call sign and the ship name.</li> </ul> <p>Existing Vessels are updated: New versions of vessel details identified by non-existing MMSI numbers, are inserted; otherwise the call sign and the ship name are updated.</p>
Validation Rules	<ul style="list-style-type: none"> <li>Vessels must be identified by IMO number.</li> <li>All the new Vessels will be activated and classified as valid upon administrator validation.</li> <li>All the Vessels – new and updated - will be classified as single hull tankers upon administrator validation.</li> </ul>
Database Transactions	<ul style="list-style-type: none"> <li>Create an Oracle directory that points to an existing path in the hosting server. The Oracle database owner must have full access to that path.</li> <li>Create an External table to map the definition of the CSV file.</li> <li>Create a stage area to allow the EIS system administrator identifies the Vessels of the CVS file to be stored on Vessels registry.</li> <li>Merge data in the operation tables that store the definition of Vessels.</li> </ul>

## 4.15 Vessel Management – OVR Synchronization

### Business process specifics

Business process specifics	
Responsibilities	<ol style="list-style-type: none"> <li>1. Update the OVR (Operational Vessel Repository) based on the newly updated CSD (CentralShipRepository) data.</li> <li>2. OVR semi-automatic update. Automatically compare the new vessel records created in EIS OVR during the last day against the vessel records stored at the MarInfo database. Differences will be stored in a stage area, an operator via the Management Console will verify and accept differences.</li> </ol>
User role(s)	<p>The programs are scheduled for execution on a daily basis.</p> <p>Update OVR is executed 1<sup>st</sup> while the semi-automatic OVR update follows.</p>
Action History	The action will be logged into the database log tables (TLOG).
Processing logic	<ol style="list-style-type: none"> <li>1. OVR update: The procedure will consider only the updated records following the last EIS OVR update.  Select all IMO+MMSI pairs from the CSD. Compare each IMO+MMSI pair with the vessel records listed in EIS OVR. <ul style="list-style-type: none"> <li>- If the OVR vessel Temporary record has null IMO and is resolved based on the IMO+MMSI pair from CSD.</li> </ul> Then if the IMO+MMSI pair is already defined as a vessel in EIS OVR then the notification reference will be updated to point to the IMO+MMSI pair vessel and the Temporary record will be deleted.  Alternatively, if the OVR vessel record is NOT resolved based on the IMO+MMSI pair from CSD it will remain as "Temporary" for the semi-automatic procedure. <ul style="list-style-type: none"> <li>- The IMO+MMSI pairs from CSD will be compared with the vessels in EIS OVR with IMO NOT NULL to identify any new vessels or update any existing valid vessel details.</li> </ul> Newly identified vessels (that is new active IMO+MMSI pairs) will be inserted in the EIS OVR. A check with the EIS OVR vessels will ensure that a pair is new. The OVR vessel flag IS_VALIDATED will be updated/defined based on the IMO+MMSI pair status in CSD.  If the status is 0 for NonActive the OVR vessel will be flagged as Temporary.  If the status is 1 for Active the OVR vessel will be flagged as Valid.  If the status is 2 for Active IMO NonActive MMSI/pair the OVR vessel will be flagged as Temporary.  The particulars of a vessel that already exists are updated based on the details of the pair from CSD; to maintain the history of changes the previous values will be kept and a new record with the new particulars is created. The comparison is based on the IMO+MMSI pair while the particulars to update include the CallSign, ShipName and vessel Flag.</li> <li>2. OVR semi-automatic update: The validation procedure automatically compares the new vessel records created in EIS OVR during the last day against the vessel records stored at the MarInfo DB.  The comparison of the EIS OVR vessel records and the MarInfo DB are based on the IMONumber if the IMONumber exists in the</li> </ol>



Business process specifics	
	<p>SSN DB. In case a record cannot be identified distinctively (e.g. compared based on MMSI when more than 1 potential entries exist from one MarInfo source), the latest entry is selected; it will be up to the user performing the manual intervention to decide if the vessel will remain unresolved or the status will change to InValid or accepted as Valid.</p> <p>Based on the comparison results the EIS OVR vessel records are updated.</p>
Validation Rules	<ul style="list-style-type: none"> <li>The validation rules for vessel identification annexed in "CS-0202 Vessel V&amp;V" are applicable.</li> </ul>
Database Transactions	<ul style="list-style-type: none"> <li>Both programs read data from the CSD.</li> <li>The OVR update modifies the values of the resolved vessel directly in the EIS tables that hold the definition and particulars of vessels.</li> <li>The semi-automatic update stored the divergent values in a table specifically defined for holding the vessel divergent particular's values.</li> </ul>

#### 4.16 EIS & STIRES interoperability – Get Enrichment data

Business process specifics	
Responsibilities	Provide STIRES with Notification and vessel specific data upon request.
User role(s)	STIRES requests EIS for the enrichment data for a given vessel.
Action History	<p>The enrichment per vessel will be done every 2 hours and prior to the sending of the Ship (AIS) notification to EIS</p> <p>Any errors are logged into the database log tables (TLOG).</p>
Processing logic	<p>STIRES requests EIS for the enrichment data by IMO Number and/or MMSI Number. At least one of the 2 must be specified and must be technically correct.</p> <p>EIS will use the IMO and/or MMSI as input variables. EIS will resolve the vessel based on the EIS OVR and will provide in response:</p> <ol style="list-style-type: none"> <li>Vessel particulars: IMO, MMSI, CallSign, Ship Name, Vessel Status + Vessel indicators: Banned and/or SHT. The vessel Flag will also be added. This should be synchronised with the upgrade of the STIRES design.</li> <li>The enrichment procedure will select per requested vessel: The ExpectedCallOfSelectedShip ShipCall with ETAToPortOfCall closer to the request timestamp. The MostRecentArrivalOfSelectedShip ShipCall ATAPortOfCall closer to the request timestamp. The latest Alert notification received by EIS.</li> </ol>
Validation Rules	<ul style="list-style-type: none"> <li>Vessels must be identified by IMO Number and/or MMSI Number.</li> <li>If the vessel is identified in EIS OVR then the vessel particulars and any notification that exists for which STIRES have access right to are provided in return.</li> </ul>

Business process specifics	
	<ul style="list-style-type: none"> <li>If the vessel is not identified in EIS OVR then a NULL record is provided in return.</li> </ul>
Database Transactions	<ul style="list-style-type: none"> <li>A vessel is resolved against the EIS OVR.</li> <li>Vessel particulars and relative relevant notifications are selected from the EIS notification specific tables.</li> </ul>

## 4.17 Voyage Calculation Process

Business process specifics	
Responsibilities	<p>Define the vessel's voyage by correlating Port, PortPlus, Hazmat and Ship notification data regarding the vessel direction and expected/actual times of departure/arrival to and from Ports.</p> <p>The primary scope of the voyage calculation is to identify all the notifications transmitted to SSN-EIS that refer to the same ship voyage. PortPlus notifications send for the same ship call are parts of the same ship voyage. However, a ship voyage may include notifications send by more than one data providers. This acceptance is based on the fact that data providers associate the PortPlus notifications for the same ship call identified by the same <i>ShipCallId</i>. A voyage may contain also notifications of type Port, Hazmat and Ship.</p>
User role(s)	SSN Core
Action History	<p>Every Port, PortPlus, Hazmat and Ship notification will be processed to calculate the voyage of the resolved vessel.</p> <p>Any errors are logged into the database log tables (TLOG).</p>
Processing logic	<p>Assumption 1: all PortPlus notifications send for the same ship call are parts of the same ship voyage.</p> <p>Assumption 2: a ship voyage may include notifications send by more than one data providers.</p> <p>Assumption 3: it is expected that for the same ShipCall - to a given PortOfCall - either a Port notification (with NextPortOfCall at ETA) or a PortPlus notification (with PortOfCall at ETAToPortOfCall) will be send but not both.</p> <p>A new voyage may begin:</p> <ul style="list-style-type: none"> <li>from the <i>LastPort</i> (LP1) at <i>ETDFromLastPort</i>, provided only in a PortPlus notificationat <i>NVL(ATAPortOfCall, ETAToPortOfCall)</i></li> </ul> <p>and end</p> <ul style="list-style-type: none"> <li>to the <i>PortOfCall</i> (PC1) at <i>NVL(ATAPortOfCall, ETAToPortOfCall)</i> provided as <i>PortOfCall</i> in the PortPlus notification</li> </ul> <p>OR anew voyage may begin:</p> <ul style="list-style-type: none"> <li>from the <i>PortOfCall</i> (LP2) at <i>ATDPortOfCall</i> provided in the PortPlus notification</li> </ul> <p>OR</p>

## Business process specifics

- from the *NextPortOfCall* (LP2) at *ETD* provided in the Port notification

and end either

- to the *NextPort* (PC2) at *ETAToNextPort* provided in the PortPlus notification

OR

- to the *NextPortOfCall* (PC2) at *ETA* provided in the Port notification, Hazmat notification and/or Ship notification.

Any notifications send during the voyage of the vessel from LP1 to PC1 will be assigned the same *VoyageId*.

The rules for assigning a new *VoyageId* or assigning a new notification to an existing voyage:

### 1. Receipt of a PortPlus notification with a new ShipCallId1.

If no Voyage exists with equal ShipCallId1 PortOfCall and NVL(ATAPortOfCall, ETAToPortOfCall) at the approximation of 1 hour then

- a new Voyage1 is defined;  
Start: LastPort at ETDFromLastPort  
End: PortOfCall and NVL(ATAPortOfCall, ETAToPortOfCall)
- a new Voyage2 is defined;  
Start: PortOfCall at ATDPortOfCall  
End: NextPort at ETAToNextPort

Receipt of a PortPlus notification with a new ShipCallId2

If Voyage2 exists with EndPort = ShipCallId2 PortOfCall and NVL(ATAPortOfCall, ETAToPortOfCall) ≈ Voyage2 ETAToNextPort with an approximation of 1 hour then ShipCall2 is assigned to Voyage2.

If Voyage2 exists and EndPort is not defined but NVL(ATAPortOfCall, ETAToPortOfCall) > Voyage2 ATDPortOfCall with an approximation of 1 hour then ShipCall2 is assigned to Voyage2.

Receipt of a PortPlus notification with a new ShipCallId3

In the exceptional case that Voyage1 exists and StartPort = ShipCallId3 PortOfCall and NVL(ATAPortOfCall, ETAToPortOfCall) ≈ Voyage1 ETDFromLastPort with an approximation of 1 hour then ShipCall3 is assigned to Voyage1.

Receipt of a PortPlus notification with a new ShipCallId4

If Voyage4 exists (see Port notification below) and ShipCallId4 LastPort = NextPortOfCall and ETDFromLastPort ≈ Voyage4ETA with an approximation of 1 hour then ShipCall4

## Business process specifics

is assigned to Voyage4.

If Voyage4 exists (see Port notification below) and ShipCallId4 NextPort = NextPortOfCall and ETAToNextPort  $\approx$  Voyage4ETA with an approximation of 1 hour then ShipCall4 is assigned to Voyage4.

If defined the *LastPort* will be the starting point. The *LastPort* is optional and as such it could be unknown if not defined. If not defined the voyage is incomplete.

The *PortofCall* attribute is mandatory and will define the ending point of 1 voyage and the starting point of another voyage if *ATDPortOfCall* is given. In case of cancellation the *PortOfCall* is cancelled hence the voyage becomes incomplete.

If defined the *NextPort* will be the ending point. The *NextPort* is optional and as such it could be unknown if not defined. If not defined the voyage is incomplete.

### 2. Receipt of Port notification.

If no Voyage exists with equal NextPortOfCall and ETA at the approximation of 1 hour then a new Voyage4 is defined.

If a Voyage2 exist and EndPort = NextPortOfCall and ETA  $\approx$  Voyage2 ETAToNextPort with an approximation of 1 hour then the Port notification is assigned to Voyage2.

If a Voyage2 exist and EndPort is not defined but ETA > Voyage2 ATDPortOfCall then the Port notification is assigned to Voyage2.

In the exceptional case that Voyage1 exists and StartPort = NextPortOfCall and ETA  $\approx$  Voyage1 ETDFromLastPort with an approximation of 1 hour then the Port notification is assigned to Voyage1.

In case of cancellation the *NextPortOfCall*, *ETA*, *ETD* are cancelled hence the voyage becomes incomplete.

### 3. Hazmat Notification. Depending on the *NextPortOfCall* and the ETA (with an approximation of 1 hour) the notification will be assigned a voyage created for a Port or PortPlus notification send before. If no such case then a new voyage will be created and assigned to the Hazmat notification.

In case of cancellation of the Port or PortPlus of the voyage the Hazmat will remain as part of the voyage. The Hazmat could/could not be updated by the data provider. In the 1<sup>st</sup> case the system will maintain the previous Hazmat in the voyage and the next Port or PortPlus notification will give the next Port of Call after the cancellation. In the 2<sup>nd</sup> case – if the data provider sends a Hazmat update – the new Hazmat will be assigned to the voyage. This way the Hazmat remains known and correlated with the ship's voyage.

In case of *NextPortOfCall* = "ZZUKN" no voyage is assigned.

### 4. Ship Notification. Depending on the *NextPortOfCall* and the ETA (with an approximation of 1 hour) the notification will be assigned a voyage created for a Port or PortPlus notification

Business process specifics	
	<p>send before. If no such case then a new voyage will be created and assigned to the Ship notification.</p> <p>In case of Ship (AIS) notification with a technically incorrect <i>NextPortOfCall</i> will not be considered in the voyage calculation.</p> <p>In case of cancellation of the Port of PortPlus of the voyage the Ship notification will remain as part of the voyage.</p> <p>In case of <i>NextPortOfCall</i>="ZZUKN" no voyage is assigned.</p> <p>Based on the time of departure from the <i>LastPort</i> and the time of arrival to the <i>PortOfCall</i> a voyage can be characterized as:</p> <ul style="list-style-type: none"> <li>- Previous Voyage: the voyage is completed and the vessel has departed from the <i>PortOfCall</i>. Criteria: current timestamp is greater than <i>ATDFromPortOfCall</i>.</li> <li>- Current Voyage: The vessel is in between the <i>LastPort</i> and the <i>PortOfCall</i>. Criteria: current timestamp in between <i>ETDFromLastPort</i> and <i>ETAToPortOfCall</i> / <i>ETA</i>.</li> <li>- Future Voyage(s): the vessel has not departed from the <i>PortOfCall</i> yet. Criteria: current timestamp is less than <i>NVL(ATAPortOfCall,ETAToPortOfCall)</i> / <i>ETA</i>.</li> </ul> <p>It is expected that the <i>LastPort</i> and the expected/actual time of departure is not notified to EIS on time; in this case a provisional indication of the <i>LastPort</i> and departure time could be given in the form of a PortPlus Notification. PortPlus notification data that originate from STIRES can be distinguished based on the data provider (being STIRES) and as such must be considered as provisional only.</p> <p>It should be noted that provisional ShipCall information shall not be provided to the request for details by a data requestor.</p>
Validation Rules	<ul style="list-style-type: none"> <li>• Vessels must be identified by IMO Number and/or MMSI Number.</li> <li>• The LASTPORT and PORTOFCALL must define a technically correct LOCATION registered in SSN-EIS.</li> <li>• LASTPORT must be different from PORTOFCALL.</li> <li>• <i>NVL(ATAPortOfCall,ETAToPortOfCall)</i> must be less than <i>ETDFromPortOfCal</i>.</li> <li>• <i>NVL(ATAPortOfCall,ETAToPortOfCall)</i> must be greater than <i>ETDFromLastPort</i>.</li> <li>• <i>ATAPortOfCall</i> must be less than <i>ATDFromPortOfCall</i>.</li> <li>• <i>ETAToNextPort</i> must be greater than <i>NVL(ATDPortOfCall,ETDFromPortOfCall)</i>.</li> </ul>
Database Transactions	<ul style="list-style-type: none"> <li>• Vessel voyage details are persisted on the VOYAGES table.</li> </ul>

---

## 5 Deployment view

---

### 5.1 Design Decisions

Since central SSN applications should support both V2 & V3 Ship MRS notification message formats for a transition period, this section describes certain design decisions to accommodate this need.

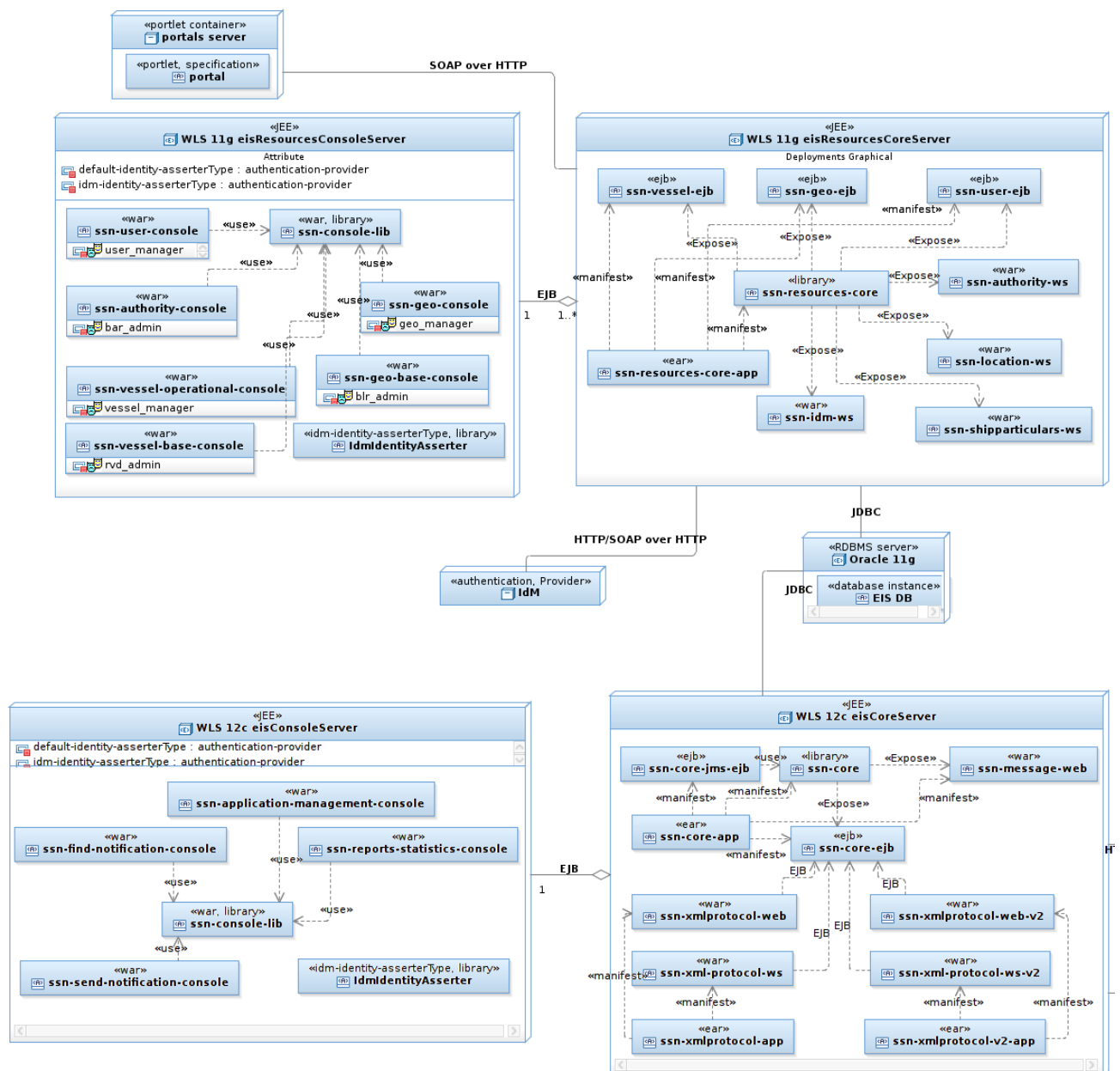
1. SSN-EIS applications (ssn-xmlprotocol-app, ssn-core-app, ssn-console) will be fully migrated to v3.
2. An additional component ssn-xml-protocol-v2 for v2 MS2SSN messages will be introduced for as long as it takes all member states to comply with v3 XML message exchange protocol.
3. Deployment diagrams below have been updated accordingly to depict changes introduced due to new application.

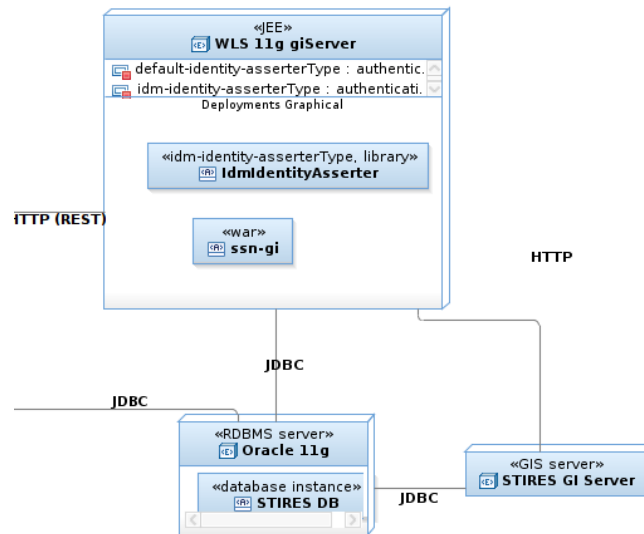
### 5.2 SSN EIS

The SSN EIS deployment view is shown in Figure 5-1

It should be noted that

- SSN EIS topology is not changed. Two JEE application servers are used for the deployment of the SSN artifacts; the first for the HTML interface (web consoles) and the second the XML (SOAP) and EJB interfaces.
- IdM system that provides the SSN SSO authentication services is out of the scope of this document.
- Identity Assertion Provider Configuration shall be done on both JEE application servers; it is related to the artifact named IdmIdentityAsserter (idm-asserter.jar).
- EMSA Portals are out of the scope of this document; the portals server included in the deployment view shows the communication path with EIS Business Services.





**Figure 5-1 SSN-EIS Deployment model**

### 5.2.1 EIS Console Server

The application server hosts the EIS consoles artifacts shall be a host running Red Hat Enterprise Linux 5.3 or later.

Technical Platform: EIS console artifacts shall be deployed onto a JEE application server (Oracle WebLogic Server 12c). The clustering feature (active/active) may be enabled.

Deployment artifacts that compose the SSNEIS web applications at runtime:

- idm-asserter: jar used for the Identity Assertion Provider Configuration;
- ssn-console-lib: war provided as library and used by all the following web console applications;
- ssn-application-management-console: war deployed to provide SSN EIS management console (SSN MC);
- ssn-send-notification-console: war deployed to provide send notifications functionality (SSN TI);
- ssn-find-notification-console: war deployed to provide find notifications functionality (SSN TI);
- ssn-reports-statistics-console: war deployed to provide SSN EIS reporting and statistics functionality.

Java Version: JDK 1.7 (1.7.55) required.

The Deployment war artifacts( ssn-application-management-console, ssn-send-notification-console, ssn-find-notification-console and ssn-reports-statistics-console) will communicate with

- EIS Core server via EJB.
- IdM system via HTTP for web users authentication.

### 5.2.2 EIS Core Server

The application server hosts EIS core artifacts shall be a host running Red Hat Enterprise Linux 5.3 or later.

Technical Platform: Authority artifacts shall be deployed onto a JEE application server (Oracle WebLogic Server 12c). The clustering feature (active/active) may be enabled.

Transaction: The SSN system is transactional, leveraging the technical platform capabilities.

Persistence: Data persistence will be addressed using the Oracle RDBMS (version 11.2.0.3) relational database that stores all data related to SSN.



Deployment artifacts that compose the SSN EIS exposed business services at runtime:

- **ssn-core-app**: ear bundles EIS core modules; it consists of
  - **ssn-core**: jar used by all the following modules; it actually implements the EIS Business Services;
  - **ssn-core-ejb**: jar deployed to expose EIS Business Services to web consoles and xmlprotocol applications as EJB (Stateless Session Beans);
  - **ssn-core-jms-ejb**: jar deployed to handle the JMS messages (Message-Driven Beans);
  - **ssn-message-ws**: war deployed to expose EIS Message Services as RESTful – XML over HTTP.
- **ssn-xmlprotocol-app**: ear bundles EIS XML protocol applications for SSN v3 schema; it consists of
  - **ssn-xmlprotocol-web**: war exposes the service for SSN EIS XML messages via HTTP(S);
  - **ssn-xmlprotocol-ws**: war exposes the service for SSN EIS SOAP messages via HTTP(S).
- **ssn-xmlprotocol-v2-app**: ear bundles EIS XML protocol applications for SSN v2 schema; it consists of
  - **ssn-xmlprotocol-v2-web**: war exposes the service for SSN EIS XML messages via HTTP(S);
  - **ssn-xmlprotocol-v2-ws**: war exposes the service for SSN EIS SOAP messages via HTTP(S).

Member states applications should be configured to send Ship Notifications via XML to different URLs, depending on the version of protocol they implement. SSN-EIS will accept post requests for

- a. V2 messages at <SSN\_HOST>/ssn-xmlprotocol-web/ssn.do
- b. V3 messages at <SSN\_HOST>/ssn-xmlprotocol-v3-web /ssn.do

Java Version: JDK 1.7 (1.7.55) required.

The Deployment artifact **ssn-core.war** will communicate with

- EIS DB via JDBC;
- IdM system via SOAP over HTTP for user management.

### 5.2.3 EIS Resources Console Server

The application server hosts EIS Resources consoles artifacts shall be a host running Red Hat Enterprise Linux 5.3 or later.

Technical Platform: EIS Resources console artifacts shall be deployed onto a JEE application server (Oracle WebLogic Server 11g). The clustering feature (active/active) may be enabled.

Deployment artifacts that compose the web applications at runtime:

- **idm-asserter**: jar used for the Identity Assertion Provider Configuration;
- **ssn-web-common**: war provided as library and used by all the following web console applications;
- **ssn-user-console**: war deployed to provide user management functionality for Operational registry;
- **ssn-authority-console**: war deployed to provide COD functionality;
- **ssn-vessel-operational-console**: war deployed to provide OVR functionality;
- **ssn-vessel-base-console**: war deployed to provide CSD functionality;
- **ssn-geo-console**: war deployed to provide Countries/ Locations / Areas management for Operational registry;
- **ssn-geo-base-console**: war deployed to provide CLD functionality.

Java Version: JDK 1.7 (1.7.55) required.

The Deployment war artifacts(**ssn-user-console**, **ssn-authority-console**, **ssn-vessel-operational-console**, **ssn-vessel-base-console**, **ssn-geo-console** and **ssn-geo-base-console**) will communicate with

- EIS Resources Core server via EJB.
- IdM system via HTTP for web users authentication.

#### 5.2.4 EIS Resources Core Server

The application server hosts EIS Resources core artifacts shall be a host running Red Hat Enterprise Linux 5.3 or later.

Technical Platform: Authority artifacts shall be deployed onto a JEE application server (Oracle WebLogic Server 11g). The clustering feature (active/active) may be enabled.

Transaction: The SSN system is transactional, leveraging the technical platform capabilities.

Persistence: Data persistence will be addressed using the Oracle RDBMS (version 11.2.0.3) relational database that stores all data related to SSN.

Deployment artifacts that exposed business services at runtime:

- `ssn-core`: war provided as library and used by all the following applications; it actually implements the EIS Business Services;
- `ssn-user-ejb`: jar deployed to expose EIS User Business Services to web consoles as EJB (Stateless Session Beans);
- `ssn-authority-ws`: war deployed to expose EIS COD Services as Web Service – SOAP over HTTP;
- `ssn-vessels-ejb`: jar deployed to expose EIS Vessels Business Services to web consoles as EJB (Stateless Session Beans);
- `ssn-ship particulars-ws`: war deployed to expose EIS Vessels Business Services as Web Service – SOAP over HTTP;
- `ssn-geo-ejb`: jar deployed to expose EIS Countries/ Locations / Areas management to web consoles as EJB (Stateless Session Beans);
- `ssn-location-ws`: war deployed to expose EIS CLD Services as Web Service – SOAP over HTTP;

Java Version: JDK 1.7 (1.7.55) required.

The Deployment artifact `ssn-core.war` will communicate with

- EIS DB via JDBC;
- IdM system via SOAP over HTTP for user management.

#### 5.2.5 SSN GI

The application server hosts `ssn-gi` artifact shall be a host running Red Hat Enterprise Linux 5.3 or later.

Technical Platform: SSN GI artifact shall be deployed onto a JEE application server (Oracle WebLogic Server 12c). The clustering feature (active/active) may be enabled.

Deployment artifacts that compose the SSN EIS web applications at runtime:

- `idm-asserter`: jar used for the Identity Assertion Provider Configuration;
- `ssn-gi`: war deployed to provide GI functionality.

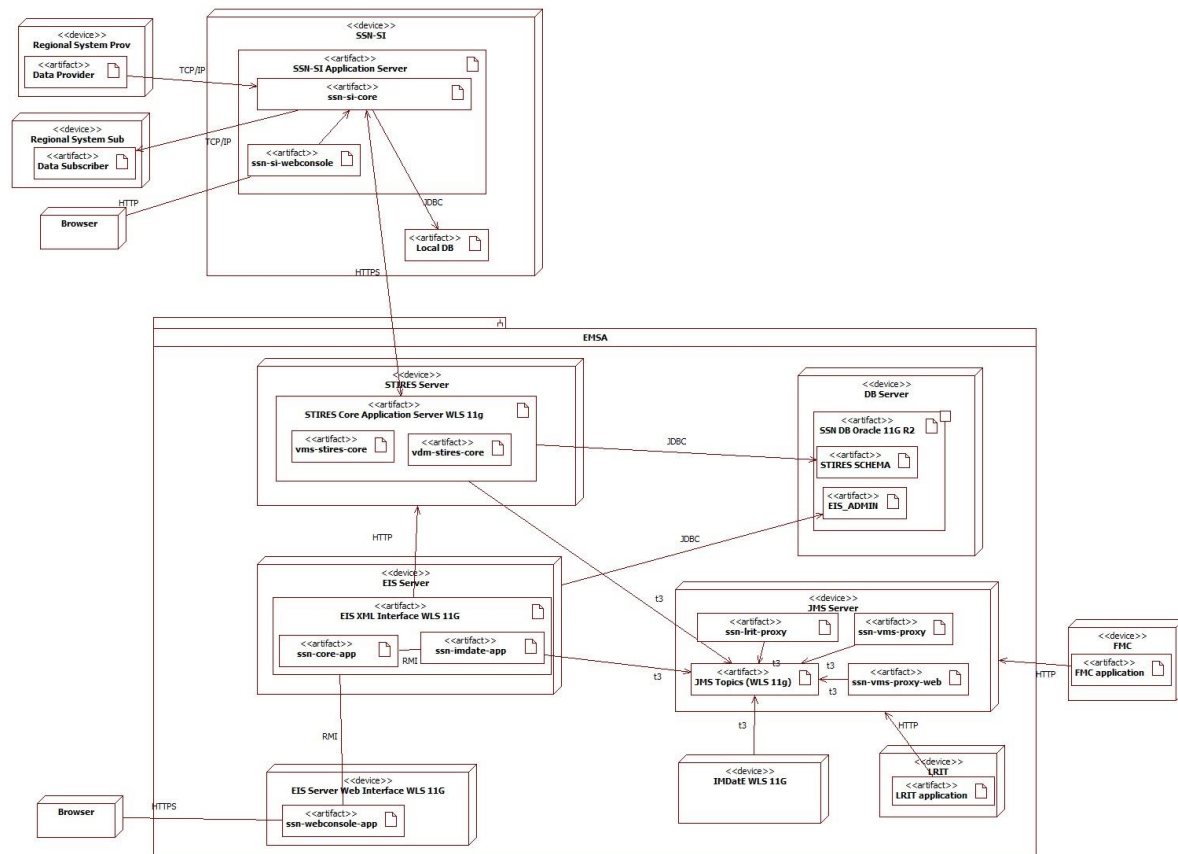
Java Version: JDK 1.7 (1.7.55) required.

The Deployment war artifact (`ssn-gi`) will communicate with

- EIS Core server via HTTP (RESTfull services).
- STIRES Database Instance via JDBC.

### 5.3 SSN-IMDaTe

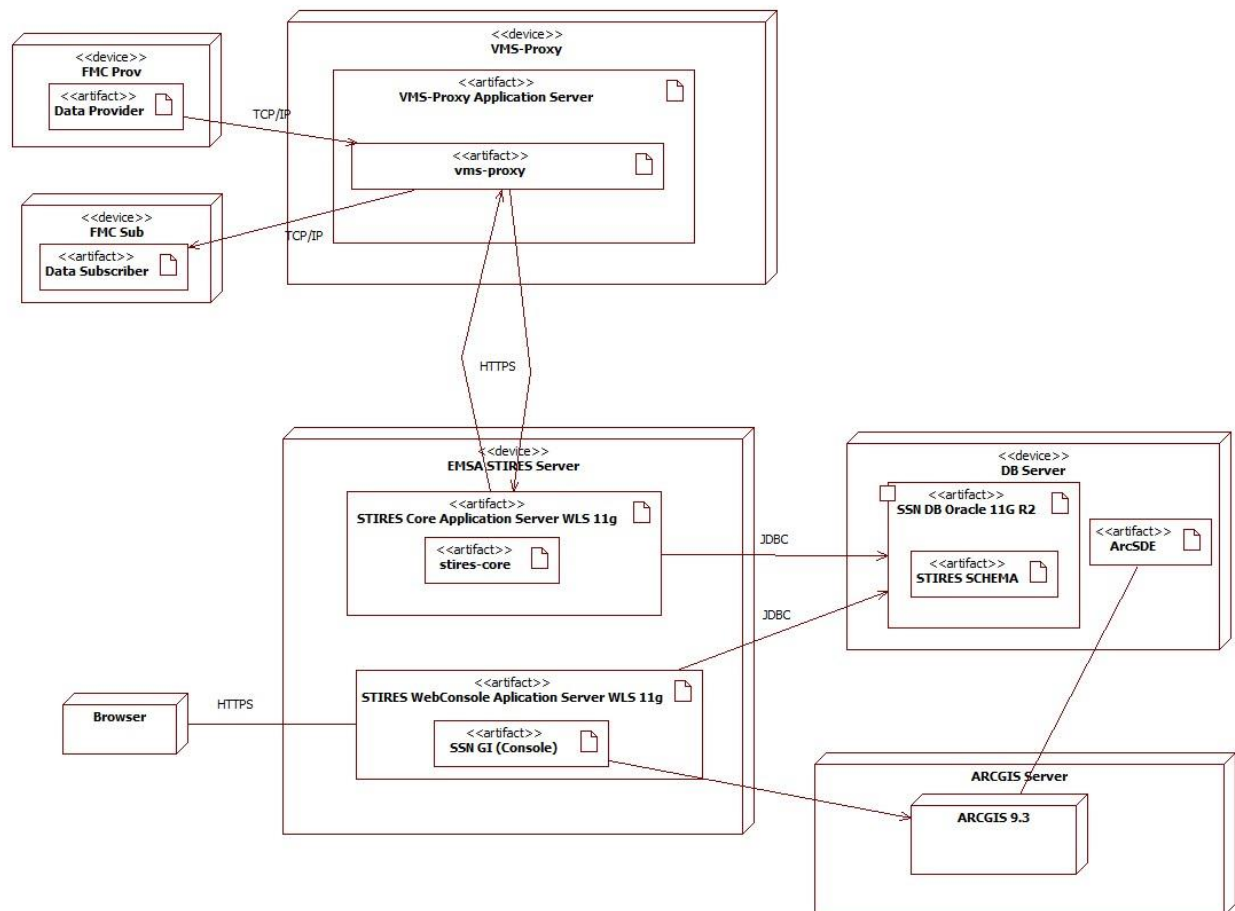
The SSN IMDaTe deployment view is presented in Figure 5-2.



SSN IMDatE is a distributed system and different components will be installed in different computer nodes. The SSN IMDatE topology comprises of the following machines:

- The Application servers are Redhat Linux machines running Enterprise Edition v. 5.0 operating system. The “Oracle WebLogic application server” is installed on these machines. The EIS, STIRES and IMDatE Proxies applications are deployed on these server.
- The Database server is a Redhat Linux machine running Enterprise Edition v. 5.0 operating system where the ORACLE RDBMS version 11g R2 is installed and it provides the data storage. The ORACLE RDBMS stores the SSN Database.

The VMS deployment view is presented in Figure 5-2.



**Figure 5-3 VMS Deployment view**

The VMS system artifacts shall be deployed onto two servers (devices) at EMSA:

- VMS-Proxy to communicate with FMCs applications;
- STIRES.

The VMS-Proxy is available for the following Operating Systems:

- Microsoft Windows XP, Server 2003.
- Red Hat Enterprise Linux 5, Centos Linux 5, Ubuntu Linux (8.04 & 9.04).

The minimum Hardware requirements are listed in the following table:

Item	Description
Processor	2.66 GHz Dual core
RAM memory	1 GB
Storage	80 GB
Network card	Gigabit network adapter
Other	Video Card, monitor, keyboard and mouse, USB ports

**Table 5-1 VMS-Proxy minimum Hardware requirements.**

Technical Platform: VMS-Proxyartifact shall be deployed onto a J2EE application server (Apache Tomcat 7.0.x – 7.0.16). The clustering feature (active/active) may be enabled.

Transaction: The VMS-Proxy system is transactional, leveraging the technical platform capabilities.

Deployment artifact that implements the VMS-Proxy application at runtime: vms-proxy-app.ear.

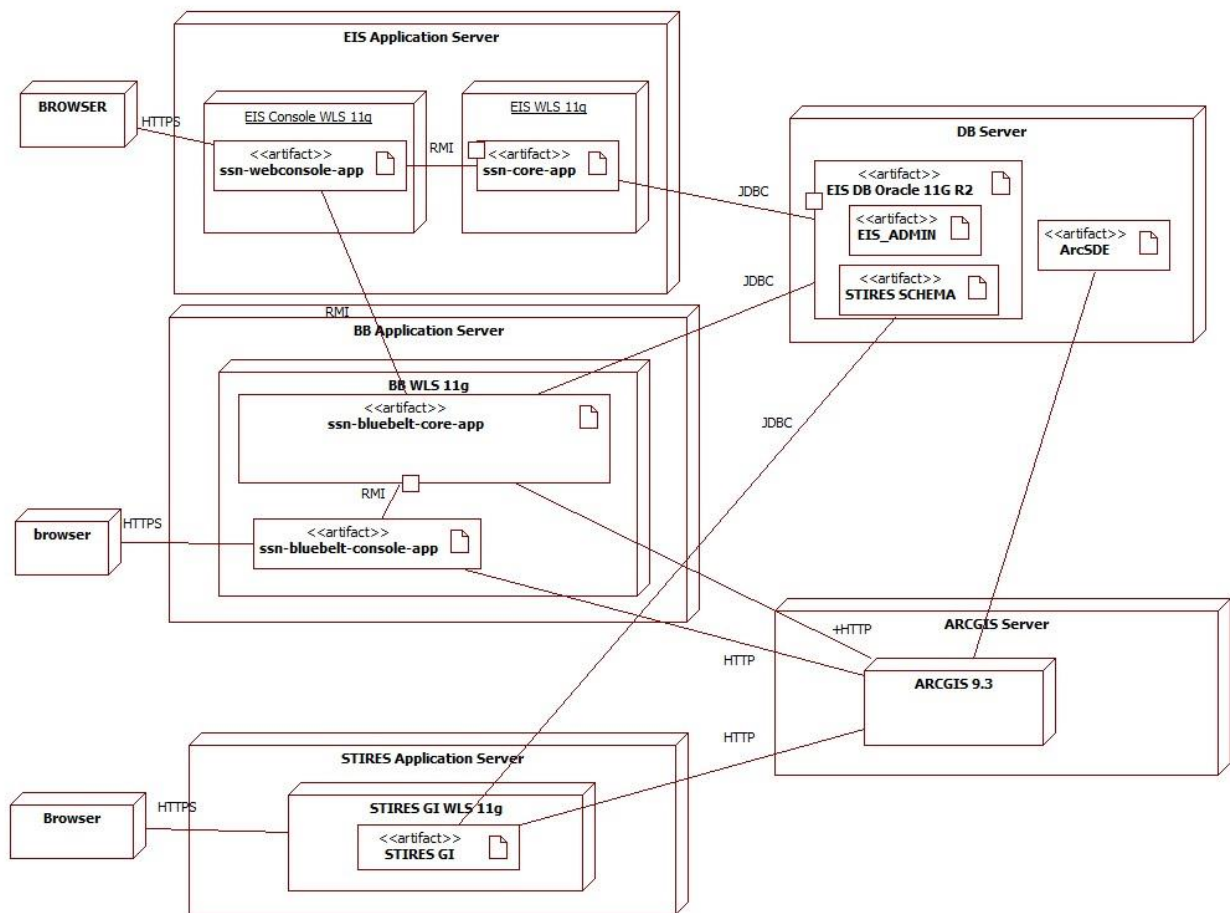
Java Version: JDK 1.6 (1.6.26) required.

The Deployment artifact vms-proxy-app.ear will communicate

- with FMCs (Data Provider and Subscribers) via TCP/IP.
- with STIRES via HTTPS.

## 5.5 SSN-Blue Belt

Deployment artifacts that compose the SSN-BB application at runtime: ssn-bluebelt-core-app.ear and ssn-bluebelt-console-app.ear in homogeneous deployment.



**Figure 5-4 SSN BlueBelt System Deployment View.**

The Deployment artifact ssn-bluebelt-core-app.ear will communicate

- with EIS DB via JDBC; two (2) Data Sources will be configured on the application server; one for EIS\_USER and a second for STIRES\_SCHEMA
- with ArcGIS server via HTTP

The Deployment artifact ssn-bluebelt-console-app.ear will communicate

- with ssn-bluebelt-core-app.ear via RMI (actually local calls due to homogeneous deployment)
- with ArcGIS server via HTTP

The update Deployment artifact ssn-webconsole-app.ear will also communicate

- with ssn-bluebelt-core-app.ear via RMI.

## Annex A: Business Rules

### Voyage Status Indicators

The following table defines the rules based on which a voyage is perceived to be at a given status at a given point in time (e.g. of the notification processing).

Status indicator	Rules
Voyage on going	<p>A. The ATDlastPort (derived from ATD portofCallPreviousVoyage) or (in case of ATDlastPort absence) the calcATDlastPort or (in case of absence of ATDlastPort or calcATDlastPort) the ETD last port is in the "past" with respect to the query timestamp in UTC, and</p> <p>B. the [ETAPortofCall+[a configurable parameter, e.g. 2 hours] is in the future with respect to query timestamp in UTC, and</p> <p>C. there is no ATA known (or in case of ATA absence) no calcATA known for the voyage.</p>
Vessel at port	<p>A. The ATDlastPort or (in case of ATDlastPort absence) the calcATDlastPort or (in case of absence of ATDlastPort or calcATDlastPort) the ETD last port) is in the "past" with respect to the query timestamp in UTC , and</p> <p>B. The ATA or (in case of absence of ATA) the calcATA is available for the voyage, and</p> <p>C. There is no ATD or calcATD known for the voyage.</p>
Voyage closed	<p>A. The ATDlastPort or (in case of ATDlastPort absence) the calcATDlastPort or (in case of absence of ATDlastPort or calcATDlastPort) the ETD last port) is in the "past" with respect to the query timestamp in UTC. and</p> <p>B. The ATD or (in case of absence of ATD ) the calcATD is available for the voyage.</p> <p>The voyage is assigned the status "Closed" which indicates that the voyage is completed.</p> <p>Closed voyages are provided in requests but are not considered in voyage consolidation.</p>
Voyage status "Unknown"	<p>We have three cases of unknown voyages.</p> <p>Case 1:</p> <p>A. The ATDlastPort or (in case of ATDlastPort absence) the calcATDlastPort or (in case of absence of ATDlastPort or calcATDlastPort) the ETD last port) is in the "past" with respect to the query timestamp in UTC, and</p> <p>B. There is no ATA or calcATA recorded for the voyage, and</p> <p>C. The [ETAPortofCall+[a configurable parameter, e.g. 2 hours] ] <u>is in the past</u> with respect to query timestamp in UTC.</p> <p>Case2:</p> <p>Any voyage which solely includes the data in a Port notification or in a v1 Hazmat / arrival from nonEU notification (that is in the voyage table record or in the notification notifying a voyage are included only the ETAPortOfCall and the ETDPortOfCall.</p> <p>Case3:</p> <p>Voyages that provide ETDlastPort and ETAPortOfCall that are both beyond the WVD and there is no ShipCallID recorded for the voyage</p>

Status indicator	Rules
	at the timestamp of the query. This relate to "future"voyages that are created by Hazmat v1 messages where the departure time and arrival time are both dummy.
Future (known) voyage	Any voyage that ETDLastPort declared in the notification and set in a future time with respect to the timestamp of the query and with an ETAPortOfCall also in the future. The ETDLastPort should not be dummy (that is it must be within the limit constraint by the WVD).
Planned call	Any voyage for which the ShipCallID is known and ETDLastPport declared in the notification is set in a future time with respect the timestamp of the query and with an ETAPortOfCall also in the future. In this case both ETDLastPort and ETAPortofCall are provided and are both beyond the WVD.  <b>Open issue: according to the Future voyage definition the ETD and ETA are dummies. In this case no voyage shall be created.</b>
Voyage cancelled	Receipt of ZZCAN for the port call reported either via PortPlus or via Port notification.
Voyage on-hold	The voyage has been created from a PortPlus notification with UpdateStatus = "U" and the original PortPlus notification (with UpdateStatus = "N") is still expected.  On-hold voyages are not provided in requests, but are considered in voyage consolidation.
Dummy voyage	Dummy voyages report an ETA to a given Port in the past; no ATA has been reported for the given Port while a later ATA is reported by a MS or detected by STIRES to another Port.  Dummy voyages are not provided in requests and are not considered in voyage consolidation.

## Voyage retrieval specific rules

The following table defines the rules of voyage retrieval that are used during the "matching" process. They are used to decide if there is a voyage in the database whose data could be correlated with the data reported in the incoming notification.

Data Provider	Notification includes	Voyage status	Which voyage has to be retrieved from the EIS database for the reported vessel (if exists) and action to be taken in case the Notification does not report the ShipCallId.
MS	ATD	Voyage closed	Fetch the most recently "closed" status voyage from the database, if exists, and initiate the voyage matching process.  If no match is to be found fetch the voyage with status "at port", if exists and try to resolve the notification with it. If no match found fetch the most recently created voyage" with status unknown and try to match. If not a match found create a new voyage in the database with the data in the notification.
STIRES	calcATD	Voyage	Fetch the most recently "closed" (ATD is reported) status voyage for the ship from the database, if exists,



		closed	<p>and check if the port of call info match with the port of call reported by STIRES. If yes register in the STIRES voyage linked with the incoming STIRES notification the voyage ID of the voyage fetched from the database (should this VoyageID is not already registered in the STIRES voyage record or should in the STIRES voyage record was registered a different voyageID).</p> <p>If a match is not found, fetch from the database the voyage of the ship with status "at port" (ATA reported, no ATD reported) if exists and repeat the matching process.. If a match is found register in the STIRES voyage linked with the incoming STIRES notification the voyage ID of the voyage fetched from the database (should this VoyageID is not already registered in the STIRES voyage record or should in the STIRES voyage record was registered a different voyageID).</p> <p>If no match is found the VoyageID already registered in the STIRES voyage shall remain un-altered). The value of the VoyageID could be NULL.</p>
STIRES	calcETDLastPort	Voyage on going	<p>Fetch the voyage with status "on going" from the database, if exists. If the LastPort data match with those in the STIRES notification (based on the rules for matching last port) assign to the STIRES voyage that is related with the notification received the VoyageID of the voyage fetched from the database. If a match is not found the VoyageID in the STIRES voyage that is related to the STIRES notification shall remain empty.</p>
MS	ATA	Vessel at port	<p>First retrieve the voyage with status "at port" in the database for the vessel, if exists. If no match is found fetch the voyage with status "on-going" and start the matching process. If no match is found fetch the most recent voyage with status unknown for the ship and start the matching process. If no match is found create a new voyage based on the data in the notification.</p>
STIRES	calcATA	Vessel at port	<p>First retrieve the voyage with status "on going" in the database for the vessel, if exists.</p> <p>If a match is found (PortofCall in the voyage fetched from the database equals with the calculated port of call from STIRES register in the STIRES voyage linked with the incoming STIRES notification the voyage ID of the voyage fetched from the database (should this VoyageID is not already registered in the STIRES voyage record or should in the STIRES voyage record was registered a different voyageID).</p> <p>If a match is not found fetch the voyage with status "at port" (ATA reported, no ATD reported) and start the matching process). If match is found update the voyage ID in the STIRES voyage. If no match is found fetch the most recent voyage with status unknown for the ship, if exists, and start the matching process</p>



			(based on LastPort and ETDLastPort??). If match is found update the voyageID in the STIRES voyage record. If no match found the voyage ID in the STIRES record shall remain un-altered.
MS	ETA Notification reports an on-going voyage (according to the on-going voyage definition)		First retrieve the voyage with status "on-going" in the database for the vessel, if exists. If no match is found fetch the voyage with status "at port", if exists, and start the matching process. If no match is found fetch the most recent voyage with status unknown for the ship and start the matching process. If no match is found create a new voyage.
MS	ETA Notification reports a future voyage (according to the future or plan voyage definition)		Check if there exists a future voyage in the database where the ETAPortOfCall is closest in the future with respect to the ETDLastPort reported in the notification and start the matching process. If no match found create a voyage.
MS	ETA Notification reports an unknown voyage with ETA in the future		Port notifications. Fetch the voyage with status "on-going" from the database, if exists, and try to resolve. If no match found fetch the most recently created voyage with status "unknown", if exists, and try to resolve the vessel. If a match is not found create a new voyage in the database.
MS	ZZCAN reported with a Portplus notification	Voyage cancelled	Change of destination. Fetch the voyage with the same ShipcallID and cancel it. <u>Be aware that the Hazmat linked to the voyage is not cancelled and should be linked with another voyage in the database.</u> That is why, following the cancellation of the ship call the system should check again the current status of voyage as reported by the notification that initially included the Hazmat information (you can retrieve this notification from the database using the Hazmat ShipcallID or the Hazmat data provider stored in the cancelled voyage record). Re-initiate the process of voyage retrieval. Based on the status of the voyage as reported in the notification and based on the processes highlighted above for retrieving a voyage from the database identify the voyage where the Hazmat info has to be linked.
MS	An update of a PortPlus message reporting Hazmat EU departure where the data provider changed the HazmatYesNo attribute value from		Retrieve the voyage with the Hazmat ShipcallID pointing to the notification that included the changed HazmatYesNo attribute change and update the voyage. In case the voyage record does not include a ShipCallID (that is it was created by the next portdata and it is not yet confirmed by the port of arrival as yet, "cancel" it completely.  Actually here the MS ask us to ignore a Hazmat EU notification previously sent with a PortPlus message – Should for the voyage next Port has been already received a Portplus from the Port of arrival the voyage

	"Yes" to No (by doing this essentially the user "cancels" the previously sent Hazmat notification		cannot be cancelled.  This is only way for a MS to report that to cancel a Hazmat attached to a Portplus message reporting EU departure.
--	---	--	--