

Country Code Base Registry

Architecture Design Document

Version 1.0

1 Table of Contents

1	TABLE OF CONTENTS	2
1	INTRODUCTION	3
1.1	PURPOSE	3
1.2	SCOPE	3
1.3	ACRONYMS AND ABBREVIATIONS	3
1.4	REFERENCES	4
1.5	DOCUMENT STRUCTURE	4
2	PROJECT STANDARDS, CONVENTIONS AND PROCEDURES	5
2.1	DESIGN STANDARDS	5
2.2	DOCUMENTATION STANDARDS	5
2.3	NAMING CONVENTIONS	5
2.4	PROGRAMMING STANDARDS	5
2.5	SOFTWARE DEVELOPMENT TOOLS	5
3	SYSTEM OVERVIEW	7
3.1	THE CCBR AS A COMPONENTS OF THE EMSA COMMON BASE REGISTRY	7
3.2	BACKGROUND	7
4	SYSTEM CONTEXT	9
4.1	EXTERNAL INTERFACE DEFINITION	9
4.2	DECOMPOSITION DESCRIPTION	10
4.2.1	CCBR Block Diagram	11
4.2.2	CCBR Layers Decomposition	26
4.2.3	CCBR Physical Diagram	27

1 Introduction

1.1 Purpose

This document provides an architecture overview of the Country Code Base Registry system.

This document has been written for the CCBR software developers or for any other member of the CMC Team in order to provide high-level technical information about the CCBR system.

1.2 Scope

EMSA has recently decided to start the implementation of Common Management Console (CMC) also known as Base registries. A Base Registry database stores reference information that is shared by several applications in the Agency. A Base Registry can be seen as a directory that uniquely identifies information items by means of a “key”. Applications using the same key can refer to the same item without any ambiguity. Some examples of information that can be shared at an Agency level are: Countries, Authorities, and port LOCODEs.

Base Registries lay down the concept of official data ownership with regard to a specific information type, providing centralized services for accessing the data which in the past was spread over several different systems.

Having this in consideration EMSA has decided to integrate the Country Code Base Registry on the ShipDB system, since it already contains some information about Countries, Maritime Identification Digits (MID) and CallSign prefixes.

1.3 Acronyms and abbreviations

The following table presents the acronyms and abbreviation used in the context of this document:

Acronyms	Description
API	Application Programming Interface
CCBR	Country Code Base Registry
CMC	Common Management Console
CUD	Create, Update & Delete
DAO	Data Access Objects
DB	Database
EMSA	European Maritime Safety Agency
ES	External Systems
HTTP	Hypertext Transfer Protocol
JAXB	Java Architecture for XML Binding
JAX-WS	Java API for XML Web Services
JDK	Java Development Kit
JSF	Java Server Faces
LRIT	Long Range Identification and Tracking of Ships
MID	Maritime Identification Digit

OSB	Oracle Service Bus
RDBMS	Relational Database Management System
RHEL	Red Hat Enterprise Linux
SGIM	Steering Group for IT and Maritime Applications
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
W3C	World Wide Web Consortium
WSDL	Web Service Description Language
XML	eXtensible Markup Language

Acronyms and abbreviation

1.4 References

The following table contains the list and reference documents:

Ref.	Title	Date	Version
[RD1]	CMC-Country Base Registry	-	V1.0
[RD2]	CCBR_Software_Requirements	14/03/2013	V0.1
[RD3]	CCBR_OSB_Interface	05/02/2013	V0.2
[RD4]	System and Application Technical Landscape	06/02/2013	V20
[RD5]	LRIT Project & Quality Management Plan	14/09/2012	V3.10
[RD6]	Note to the attention of the SGIM Members	04/07/2012	-

Reference Documents

1.5 Document Structure

The document contains the following parts:

1. Introduction, which includes the project scope and the purpose, acronyms, references and structure of the document.
2. Project standards, conventions and procedures, which briefly describes design, documentation and programming standards as well as naming conventions and software development tools.
3. The CCBR Overview, which briefly introduces the context and the main system components and functionalities.

2 Project Standards, Conventions and Procedures

2.1 Design standards

The CCBR application design follows the object-oriented methodology.

The UML (Unified Modelling Language) has been adopted to model the design of the application.

2.2 Documentation standards

In what concerns their appearance, all the documents produced within the CCBR project use the same template. Microsoft Word is the tool used to produce this document (as well as the rest of the documentation).

2.3 Naming conventions

The conventions that will be used for naming files, programs, modules and other structures will be the ones described in [RD 5] and [RD 6].

2.4 Programming standards

The CCBR will be implemented having always in consideration the industry standards, including:

- EMSA SGIM Standards
- ICT Standards
- Performance standards;
- Industry standard virus checking and network security procedures on all messages to prevent malicious attacks (e.g., Structured Query Language (SQL) injection);
- Standard user management and user authentication mechanism;
- Standard secure internet based communication links for interfacing with other systems or between CCBR components (standard communication protocols);

2.5 Software development tools

The CCBR will be developed using the following software:

- Microsoft Window 7 OS
- Liferay Portal 5.2 SP5
- Oracle Service Bus 11.1.1.3 on Oracle Weblogic 10.3.3.0
- Oracle weblogic 10.3.6 as the CCBR Application Server
- Oracle weblogic 10.3.3.0 as the ShipDB Application server
- NetBeans 6.9.1 for the Development of the ShipDB module.
- Oracle JDeveloper 11g 11.1.1.6.0 for the development of the OSB services and Liferay Portlets

- Java Development Kit (JDK) 1.6.0.25
- Hibernate 3 for the database access.
- Oracle Database 11.2.0.1.0

The CCBR will be developed using the following technologies:

- Simple Object Access Protocol (SOAP)
- eXtensible Markup Language (XML)
- Java Architecture for XML Binding (JAXB)
- Java API for XML Web Services (JAX-WS)

In the final environment, the CCBR application will be deployed on an Oracle Weblogic application server. The system will run on Red Hat Enterprise Linux (RHEL) servers with an Oracle 11g database.

The exchange of messages with other components (OSB) is based on Simple Object Access Protocol (SOAP) and eXtensible Markup Language (XML).

3 System Overview

3.1 The CCBR as a components of the EMSA Common Base Registry

The CCBR may be viewed as a component of the EMSA common Base Registries. It will allow storing information about Countries and associated MID and Call Signs.

Once this project is completed, the other Maritime applications will have the possibility to retrieve country information and the associated MID and Call Signs from the Country Code Base Registry system. The Country Code Base Registry will also allow the Human Users to view the Country information through a User web interface.

3.2 Background

A CUR-AM Working Group was set-up in June 2012 to address:

- The assessment of existing tools and practices to handle registries for LOCODES, Authorities and Areas within the EMSA maritime applications;
- The requirements for common Base Registries definition;
- The functional requirements for a Common Management Console (CMC);
- The potential way forward for implementing a harmonized approach within the framework of the IMDatE project;

The WG has defined a Base Registry database, which should store reference information that is shared by several applications in the Agency. A Base Registry can be seen as a directory that uniquely identifies information items by means of a “key”. Applications using the same key can refer to the same item without any ambiguity. Some examples of information that can be shared at an Agency level are: Countries, Authorities, and port LOCODEs.

The Base Registry lays down the concept of official data ownership with regard to a specific information type, providing centralized services for accessing the data which in the past was spread over several different systems.

The WG suggested in a Report on the existing practice (Report #4) that the first Base Registries to be implemented are the following:

- Countries
- Authorities
- LOCODEs
- Geographical Areas

The CUR-AM WG report was submitted for discussion in late December during the SGIM meeting where it was decided to develop:

- a common console management and common Geographical Area Base Registry under the framework of the IMDatE contract;
- **a common Country Code Base Registry to be done under the framework of the LRIT contract;**
- common Authorities and LOCODE Base Registries to be done under the framework of the SSN contract;

4 System Context

The CCBR should be implemented in order to store country reference information that is shared by several applications in the Agency. The benefits of having such system are to maintain a good quality of data on the different EMSA Maritime Applications, by having only one source of information, and in long term, reducing the cost of the maintenance of EMSA Maritime Applications.

The list of EMSA Maritime Applications that may use the CCBR system is already pre-defined:

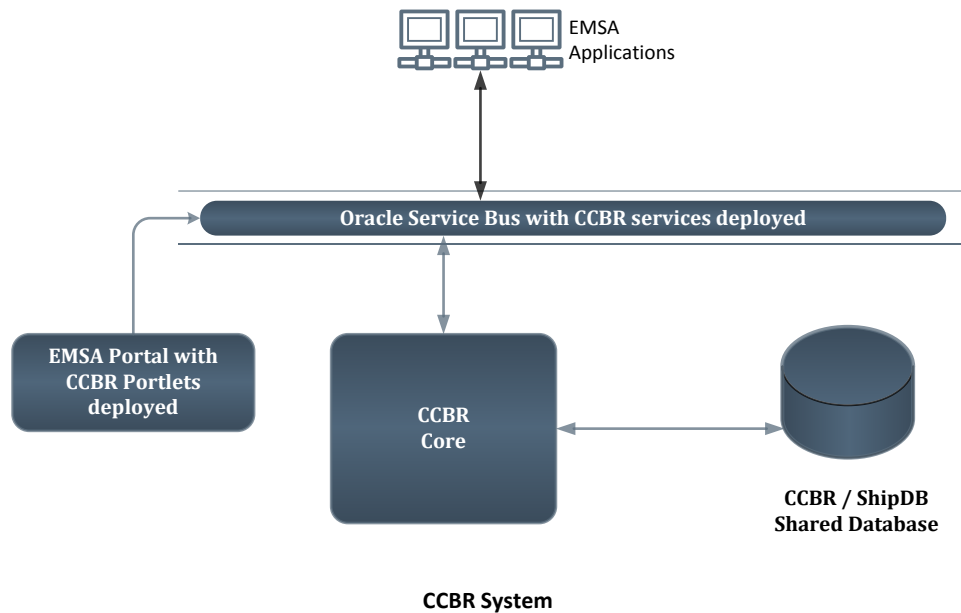
Application	appAcronym
THETIS	THETIS
SafeSeaNet	SSN
CleanSeaNet	CSN
STCW-IS	STCW
RuleCheck	RCHECK
MaKCs	MAKCS
LRIT DC	LRITDC
LRIT I&B	LRITIB
LRIT IDE	LRITIDE
LTIT Ship DB	LRITSDB
IMDatE	IMDATE
MARSURV	MSURV
MARSURV 2	MSURV2
Country Code Base-Registry	CCBR

EMSA Maritime Applications

4.1 External interface definition

The CCBR overall system is composed by three different modules:

- CCBR Core
- CCBR Portlets
- CCBR OSB Services



Each of the components will interface with the other components in a different way.

The **CCBR Core** will interface with the *CCBR/ShipDB shared database* in order to store information related with its own operation functionalities (i.e. store logging data, store users subscription data, user notification data, etc.) and to collect existent data to be used during operations (i.e. retrieve data about countries, MID's and Callsign's, notifications, user subscriptions, operations performed by users, etc.). The **CCBR Core** will also interface with the CCBR services deployed on the Oracle Service Bus platform. Through this interface, the CCBR Core will be able to notify the users about modifications in the CCBR and receive from them request to provide CCBR stored data.

The **CCBR OSB Services** will be deployed in the EMSA Oracle service Bus platform and will expose the CCBR service to the other EMSA Maritime Applications, including the EMSA Portal. By using this interface, the EMSA Maritime Applications will be able to request data stored in the CCBR. This interface and the supported operations are described in the [RD3] document. In order to provide the described services, the *CCBR OSB Services* will interact with the *CCBR Core*.

The **CCBR portlets** will be deployed in the EMSA Liferay Portal and will interface with the *CCBR OSB Services* in order to request CCBR data to be displayed in a web interface (portal).

It's not described in this section as an interface, but the ShipDB system will be responsible for feeding the *CCBR/ShipDB shared database* with data about countries, Mid's and Callsign's and to control the CUD operations performed over that information.

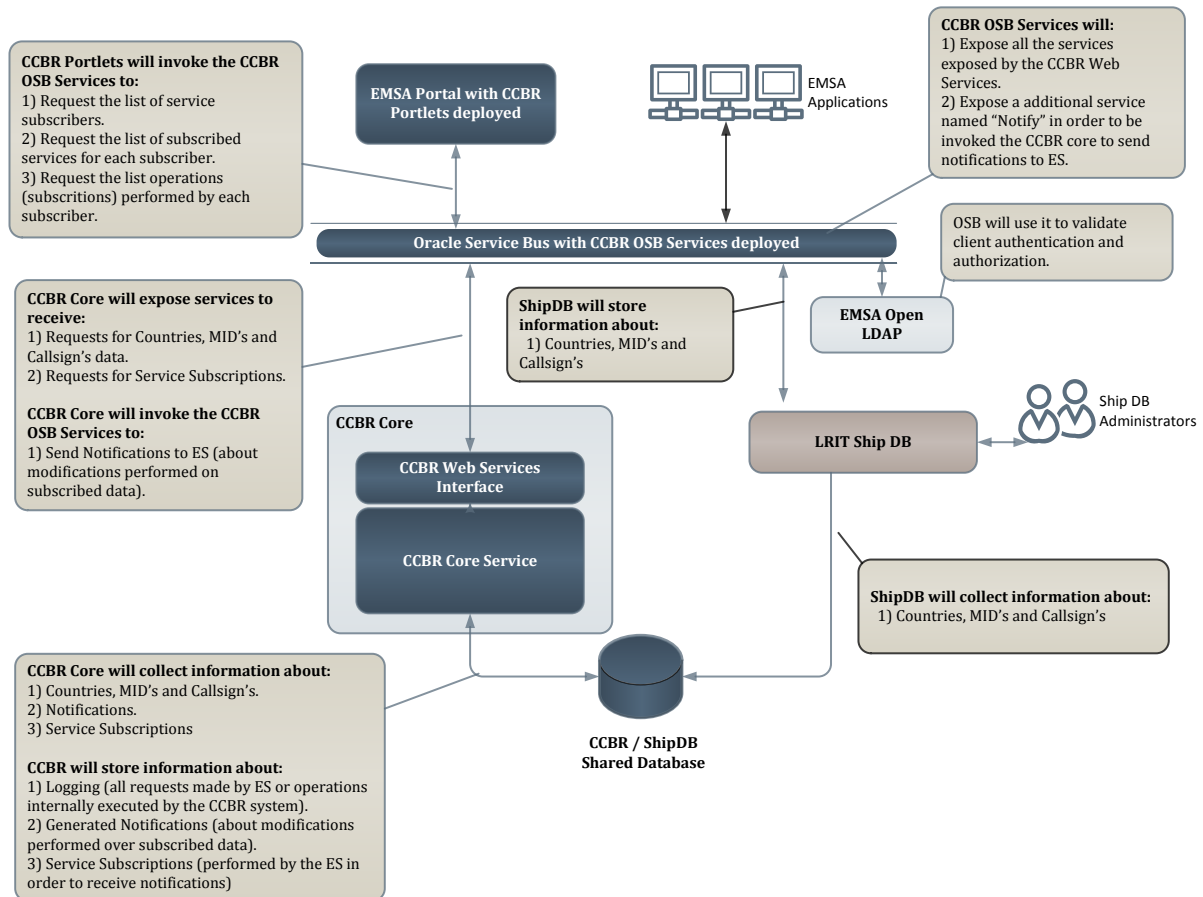
4.2 Decomposition description

In this section two different approaches to the CCBR architecture description will be followed. In the first one, the CCBR will be described as a set of block diagrams representing the different functional

modules or sub-systems. The second approach describes the CCBR as a set of different layers organized in a way that each one provides support and base functionalities for the other layers.

4.2.1 CCBR Block Diagram

The general composition (top level block diagram) of the CCBR is illustrated in the following figure:



CCBR Block Diagram Architecture

The CCBR overall system may be detached in 3 different modules: CCBR Core, CCBR Portlets and CBR OSB Services.

When a request is made from the *EMSA Applications (ES)* or from the *CCBR Portlets* to the *CCBR OSB Services*, the *CCBR OSB Services* will forward that request to the *CCBR Core* in order to be processed. The *CCBR Core* will process the received request and provide the results when it completes its processing (synchronous model). During the processing, the *CCBR/ShipDB Shared database* will be queried in order to retrieve the needed information.

Periodically, the *CCBR Core* will be searching for modifications in the CCBR data (the modifications are performed by the *ShipDB application*). If a modification is detected, the *CCBR Core* will prepare a notification message and delivery it to the CCBR service subscribers (*EMSA Applications*) through the *CCBR OSB Services*.

4.2.1.1 CCBR Portlets

The **CCBR Portlets module**, as the name says, is a set of portlets providing a specific service. Through them, the end users will be able to visualize information about the CCBR services. Together they will form the CCBR web interface.

Portlets are pluggable user interface software components that are managed and displayed in a web portal. Portlets produce fragments of markup code that are aggregated into a portal. Typically, following the desktop metaphor, a portal page is displayed as a collection of non-overlapping portlet windows, where each portlet window displays a portlet. Hence a portlet (or collection of portlets) resembles a web-based application that is hosted in a portal. In the CCBR application case, the portlets will be deployed in the EMSA Liferay Portal solution.

Liferay Portal is a free and open source enterprise portal written in Java and distributed under the GNU Lesser General Public License and proprietary licenses. It is fundamentally constructed of functional units called portlets. Liferay is sometimes described as a content management framework or a web application framework. Liferay's support for plugins extends into multiple programming languages. Liferay Portal is Java based and runs on any computing platform capable of running the Java Runtime Environment and an application server. Liferay is available bundled with a servlet container. In the CCBR application case the portlets will be developed using Java and the portlets servlet container will be the Weblogic.

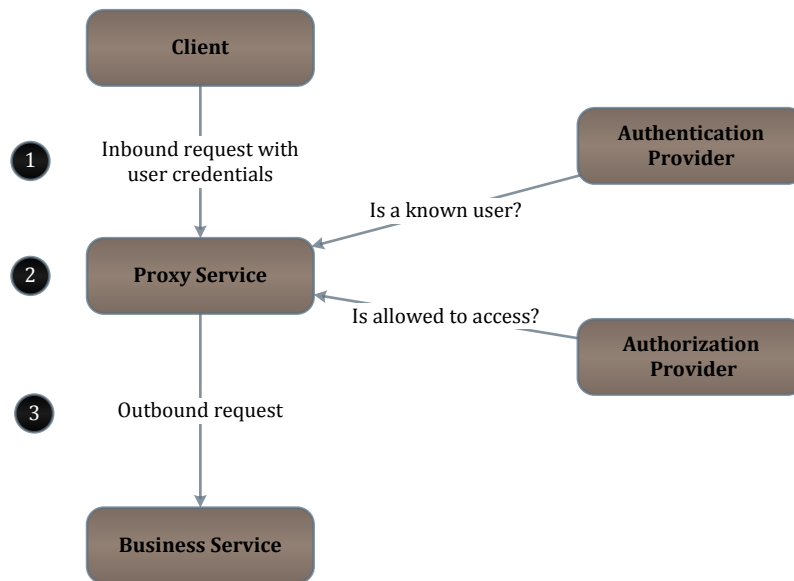
4.2.1.2 CCBR OSB Services

The **CCBR OSB Services module** will be a set of synchronous web-services exposing some of the CCBR services to a pre-defined list of clients (EMSA Maritime Applications). Those web-services, developed using Java technology, will be deployed in the EMSA Oracle Service Bus platform.

Since not all the clients may have access to the system, the CCBR OSB services will be protected using Basic Authentication. This level of security ensures that Oracle Service Bus proxy services handle only the requests that come from authorized clients.

The following figure illustrates how user identities flow through Oracle Service Bus when the OSB is configured as follows:

- Require clients to provide user names and passwords in their requests
- Authenticate clients



CCBR OSB Services Security

The illustration begins with the inbound request and ends with the outbound request:

1. A client sends a request to a proxy service. The request contains the user name and password credentials.
2. The proxy service asks the domain's authentication provider if the user exists in the domain's authentication provider store. If the user exists, the proxy service asks the domain's authorization provider to evaluate the access control policy configured for the proxy service.
3. If the proxy service's access control policy allows the user access, the proxy service processes the message and sends it to the business service.

In the CCBR case, the Authentication Provider will be an *LDAP Authentication Provider* accessing to an external Open LDAP store. Oracle Weblogic Server is already prepared to provide LDAP authentication using Open LDAP.

The **CCBR OSB Services module** will be composed by 4 (four) different components:

- CCBSubscription
- CCBInformation
- CCBManageCountries
- CCBNotification

And each component may be divided into 2 sub-components: a **Proxy Service** and a **Business Service**.

The *Proxy Services* are definitions of intermediary Web services that Oracle Service Bus implements and hosts locally. Oracle Service Bus uses proxy services to route messages between business services (such as enterprise Web services and databases) and service clients (such as presentation applications or other business services). A proxy service configuration includes its interface, transport settings, security settings, and a message flow definition. The message flow definition

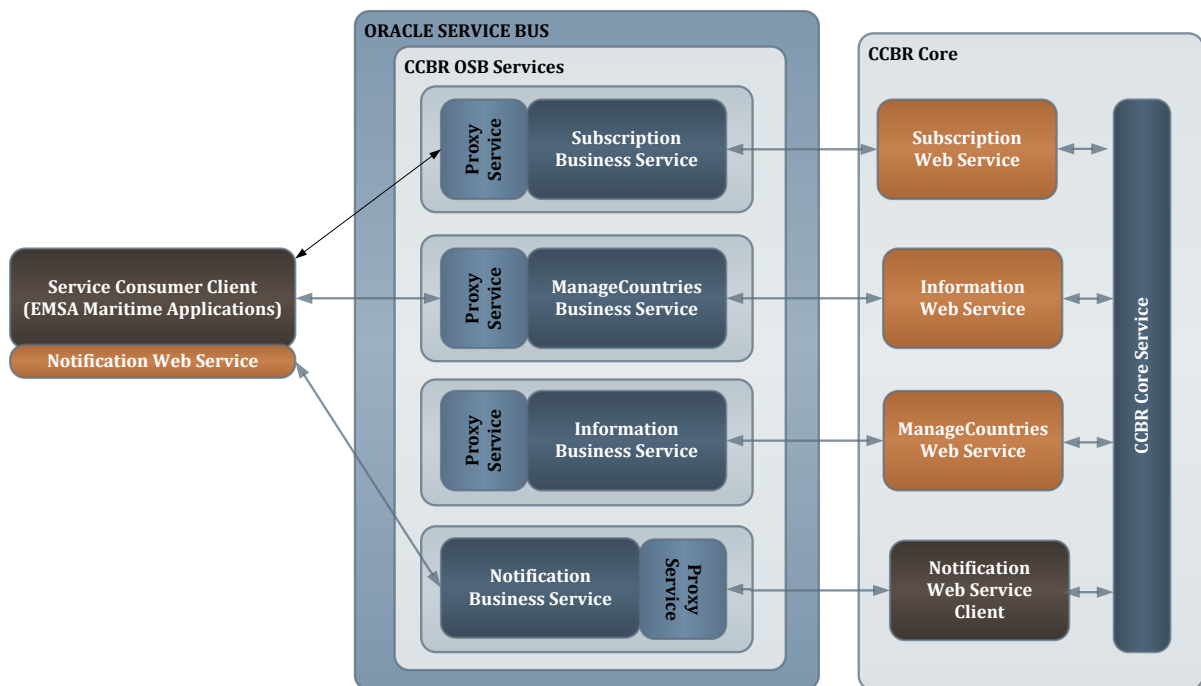
defines the logic that determines how messages are handled as they flow through the proxy service. If a proxy service is based on a WSDL document, the configuration also includes a WSDL port or a WSDL binding.

The *Business Services* are definitions of enterprise Web services to which Oracle Service Bus is a client. Those external Web services are implemented in and hosted by external systems, so Oracle Service Bus must know what to invoke, how to invoke it, and what to expect as a result. Oracle Service Bus business services model those interfaces so that Oracle Service Bus can invoke the external services. A business service configuration includes its interface, transport settings, and security settings. (It does not include a message flow definition.) If the business service is based on a WSDL, the configuration also includes a WSDL port or a WSDL binding.

In the CCBR application, the Service Consumer Clients (EMSA Maritime Applications) will invoke the CCBR Subscription, Information and ManageCountries Proxy Services (intermediary Web services) in order to request the needed CCBR information. Those requests will then be forward to the respective Business Services in order to be sent to the CCBR Core to be processed.

The same flow will occur in the opposite direction for the CCBR Notifications. In this case, the CCBR Core will be the Service Consumer and will invoke the CCBRNotification Proxy Service in order to send the Notifications. Those requests will then be forward to the respective Business Service in order to be processed and forward to the different EMSA Maritime applications.

The following figure presents the flow described above:



CCBR OSB Services

The **CCBRSubscription component** will be responsible for extending the services made available by the CCBRSUBSCRIPTION Web service implemented by the CCBR Core.

This component will expose the following operations:

Service Operation	Service Description
signSubscription(String appAcronym, String announcementEndpoint)	By calling this WebMethod an external system (Application identified by appAcronym) signs for the subscription service.
updateSubscription(String appAcronym, String announcementEndpoint)	By calling this WebMethod an external system (Application identified by appAcronym) updates his web service announcement endpoint.
revokeSubscription(String appAcronym)	By calling this WebMethod an external system (Application identified by appAcronym) revokes the subscription service.
Subscribe(String appAcronym , String subscriptionType, String subscriptionCode)	By calling this WebMethod an Application subscribes to the announcement service for the: Country identified by alpha2code. Regional Agreement identified by the Regional Agreement Code. Country Type identified by the country Type Code.
Unsubscribe(String appAcronym , String unsubscriptionType, String unsubscriptionCode)	By calling this WebMethod an Application unsubscribes to the announcement service for the: Country identified by alpha2code. Regional Agreement identified by the Regional Agreement Code. Country Type identified by the country Type Code.

getServiceSubscribers()	Returns the list of subscribed external systems.
getSubscriptionOperations(String appAcronym, javax.xml.datatype.XMLGregorianCalendar startDate, javax.xml.datatype.XMLGregorianCalendar endDate)	Returns the list of the operations performed by an external system (Application identified by appAcronym) matching the search criteria (between the startDate and endDate including boundaries). If no dates are provided, this WebMethod should return the full list of operations.
getServiceSubscribersList (String alpha2code)	Returns the list of subscribed external systems for a specific country. The service subscribers presented in this list will receive an announcement about an operation (insert, update or delete) performed over a country.
updateServiceAnnouncementStatus(String appAcronym, String alpha2code, String announcementType, javax.xml.datatype.XMLGregorianCalendar operationDate)	Since the service announcement is handled by the CCBR OSB Services, this method is used to store in the database information about the announcement sent to the ES.

The **CCBRManageCountries component** will be responsible for extending the services made available by the CCBRManageCountries Web service implemented by the CCBR Core.

This component will expose the following operations:

Service Operation	Service Description
insertCountry(Country cntr)	Insert a new country in the central database.
updateCountry(Country cntr)	Updates the data of an existent country in the central database.
deleteCountry(String alpha2Code)	Deletes an existent country from the central database
insertMID(MID mid)	Insert a new MID in the central database
updateMID(MID mid)	Updates the data of an existent MID in the central

	database.
deleteMID(String midID)	Deletes an existent MID from the central database
insertCallsign(Callsign callsign)	Insert a new Callsign in the central database
updateCallsign(Callsign callsign)	Updates the data of an existent Callsign in the central database.
deleteCallsign(String callsignIdD)	Deletes an existent Callsign from the central database

The **CCBRInformation component** will be responsible for extending the services made available by the CCBRIInformation Web service implemented by the CCBR Core.

This component will expose the following operations:

Service Operation	Service Description														
getCountryList(CountryDetailedSearchCriteria searchMsg)	Returns the list of countries matching the search criteria and respective details.														
getCountryFullList()	Returns the full list of countries and respective details.														
getSubscribedCountryList(String appAcronym)	Returns the details of the Countries subscribed by the requestor.														
getCountry(CountrySearchCriteria searchMsg)	<p>Returns the details of the Country identified by a given search parameter.</p> <p>Inside this message, one of the following search parameters must be provided in order to perform the search:</p> <table border="1"> <thead> <tr> <th>Parameter</th><th>Meaning</th></tr> </thead> <tbody> <tr> <td>NumericCode</td><td>Returns the country identified by a given 'numericCode'.</td></tr> <tr> <td>Alpha2Code</td><td>Returns the country identified by a given 'alpha2Code'.</td></tr> <tr> <td>Alpha3Code</td><td>Returns the country identified by a given 'alpha3Code'.</td></tr> <tr> <td>Callsign</td><td>Returns the country identified by a given 'callsign'.</td></tr> <tr> <td>MID</td><td>Returns the country identified by a given 'MID'.</td></tr> <tr> <td>LRIT_ID</td><td>Returns the country identified by a given 'lritId'.</td></tr> </tbody> </table>	Parameter	Meaning	NumericCode	Returns the country identified by a given 'numericCode'.	Alpha2Code	Returns the country identified by a given 'alpha2Code'.	Alpha3Code	Returns the country identified by a given 'alpha3Code'.	Callsign	Returns the country identified by a given 'callsign'.	MID	Returns the country identified by a given 'MID'.	LRIT_ID	Returns the country identified by a given 'lritId'.
Parameter	Meaning														
NumericCode	Returns the country identified by a given 'numericCode'.														
Alpha2Code	Returns the country identified by a given 'alpha2Code'.														
Alpha3Code	Returns the country identified by a given 'alpha3Code'.														
Callsign	Returns the country identified by a given 'callsign'.														
MID	Returns the country identified by a given 'MID'.														
LRIT_ID	Returns the country identified by a given 'lritId'.														
getParentCountry(String alpha2Code)	Returns the details of the Parent Country of the Country identified by a given 'alpha2Code'.														

getChildCountryList (String alpha2Code)	Returns the details of the Child Countries of a Country identified by a given 'alpha2Code'.
getCountryMIDList(String alpha2Code)	Returns all the MID's associated with a Country identified by a given 'alpha2Code'.
getCountryCallsignList(String alpha2Code)	Returns all the Callsign's associated with Country identified by a given 'alpha2Code'.
getCountryFlagImageList()	Returns the flag images (small, medium, large) of all Countries.
getCountryFlagImages(String alpha2Code)	Returns the flag images (small, medium, large) of a Country identified by the given 'alpha2Code'.
getCountryFlagImage(String alpha2Code, Integer imageType)	Returns a specific flag image size of a Country identified by the given 'alpha2Code'.
getCountryFlagImageLocation(String alpha2Code, Integer imageType)	<p>Returns the full location (URL) of the flag image with a specific size (small, medium or large) of a specific Country.</p> <p>Ex:</p> <p>If the provided imageType is 0:</p> <ul style="list-style-type: none"> ➤ http://CCBR/images/countryFlags/PT_S.png <p>If the provided imageType is 1:</p> <ul style="list-style-type: none"> ➤ http://CCBR/images/countryFlags/PT_M.png <p>If the provided imageType is 2:</p> <ul style="list-style-type: none"> ➤ http://CCBR/images/countryFlags/PT_L.png
getCountryFlagImagesLocationContext()	<p>Returns the context (URL) of the country flag images location.</p> <p>Ex:</p> <p>The context is : http://CCBR/images/countryFlags/</p> <p>The client should add to the URL the flag image name in order to retrieve the image itself. The flag image name is composed by : "<alpha2Code>"+ " " + "<imageSize>"+ ".png"</p> <p>Note: the ImageSize may assume the following values: "S" "M" or "L".</p>
getCountryHistory(HistorySearchCriteria)	Returns the history country information in a specific time

searchMsg)	or a period of time start and end dates are included in the boundaries when present
getCountryTypes	Returns a list with the existent Country Types.
getCountryCategories	Returns a list with the existent Country Categories.
getSubscribedCountries	Returns only Countries subscribed by a specific external system.
getSubscribedRegionalAgreements	Returns only Regional Agreements subscribed by a specific external system.
getSubscribedCountryTypes	Returns only the Country Types subscribed by a specific external system.

The CCBNotification component will be responsible for implementing a Web Service to be used by the CCB Core in order to deliver the CCB Notifications to the different EMSA Maritime Application.

This component will expose the following operation:

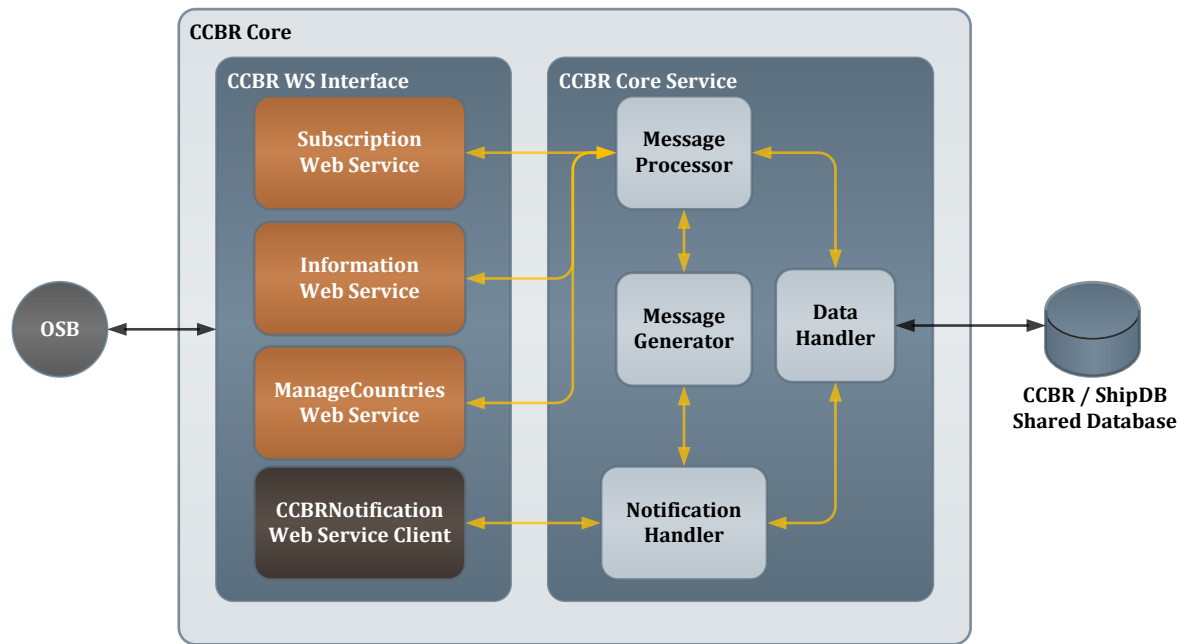
Service Operation	Service Description
notify(SubscriberNotification notification)	Web Method to be called by the Country Base Registry in order to deliver a change notification to an External System.

4.2.1.3 CCB Core

The **CCBR Core module** will be developed using Java technology and is the central CCB module. It will be composed by two different components:

- *CCBR WS Interface*
- *CCBR Core Service.*

The following figure presents the high-level architecture for this module:



CCBR Core Module high-level architecture

The **CCBR WS Interface** implements three different Web Services: the *Subscription*, *ManageCountries* and the *Information* Web Service, and one Web Service client: the *CCBRNotification*.

The three mentioned web services will be responsible for:

- Handling with the CCBR Service subscription.
- Managing the Countries and associated information.
- Returning information about the Countries and associated information.

The web service client will be responsible for:

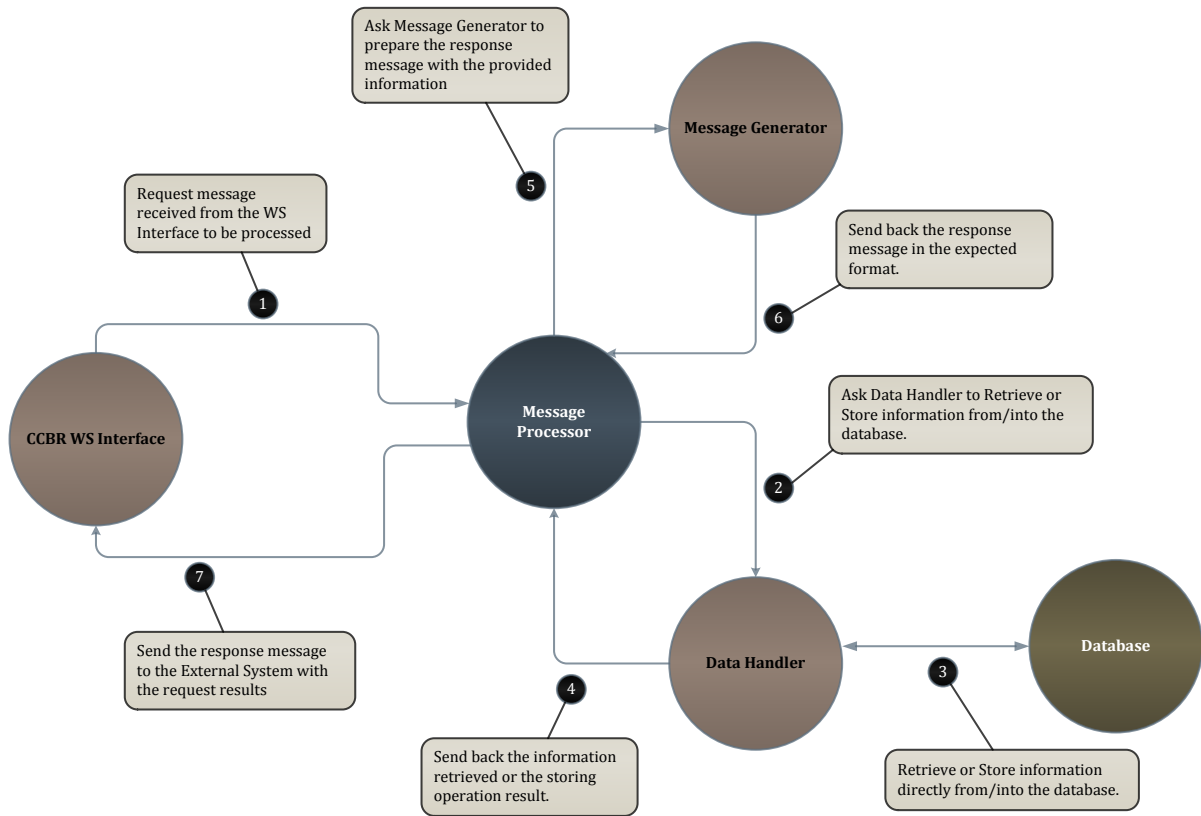
- Invoking a web service in the OSB side in order to send service notification messages to the different client applications.

The **CCBR Core Service Component** may be detached in 4 different sub-components:

- Message Processor
- Notification Handler
- Message Generator
- Data Handler

The **Message Processor** will be responsible for processing all the request messages received from the CCBR WS Interface. It will use the *Data Handler* to retrieve or store the needed data from or to the database and use *Message Generator* to build the response message.

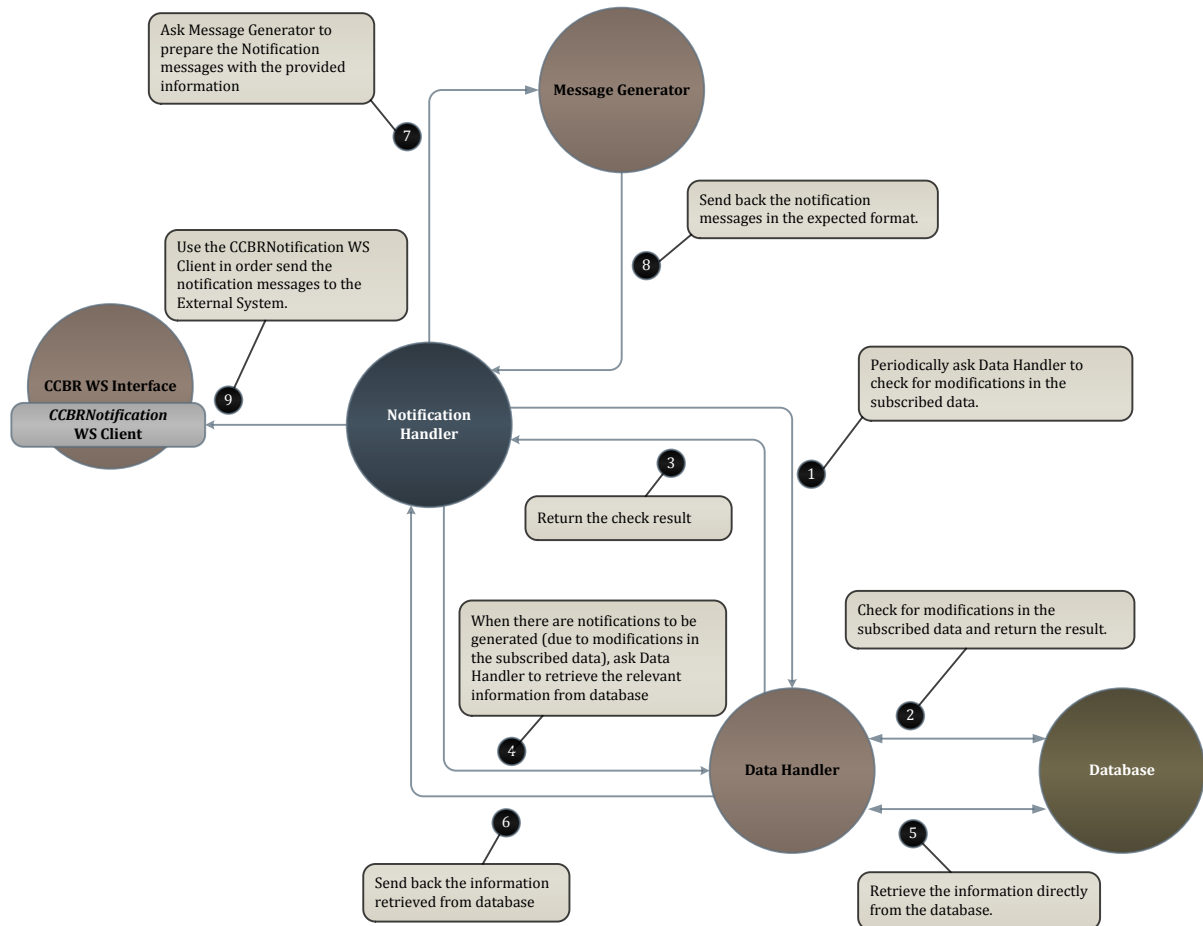
The following diagram presents the described flow:



Message Processor Data Flow

The **Notification Handler** will be responsible for detecting situations when a notification needs to be generated and sent to the EMSA Maritime application in order to inform them about modifications in the subscribed data. In order to detect those changes, the *Notification Handler*, will periodically read information from the database (by using the *Data Handler*) and check if there are notifications to be processed. In the affirmative case, the *Notification Handler* will use the *Data Handler* to retrieve the needed data from the database and use the *Message Generator* to generate the notifications to be sent. As soon as the notifications are prepared to be sent, the *Notification Handler* will use the *CCBRNotification Web Service Client* to deliver them to the external systems.

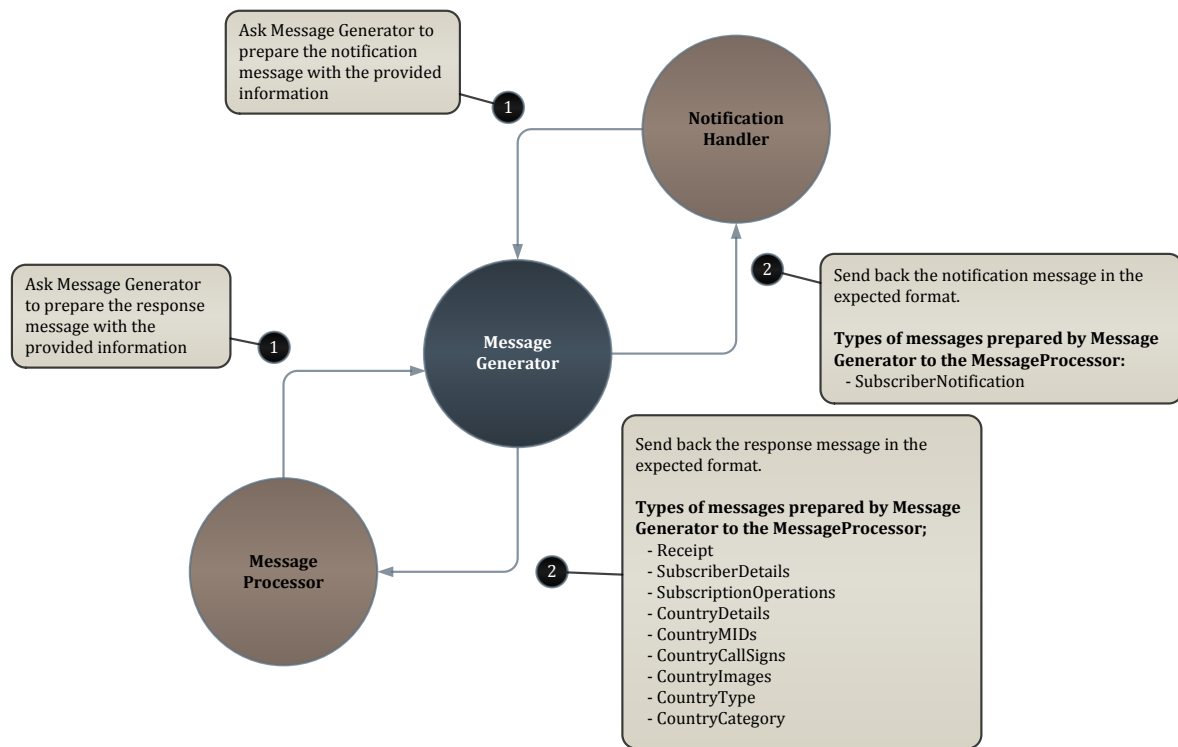
The following diagram presents the described flow:



Notification Handler Data Flow

The **Message Generator** will be responsible for generating the messages in the correct format (accordingly with the defined schemas). It will receive requests from the *Message Processor* and from the *Notification Handler* and provide the appropriate response.

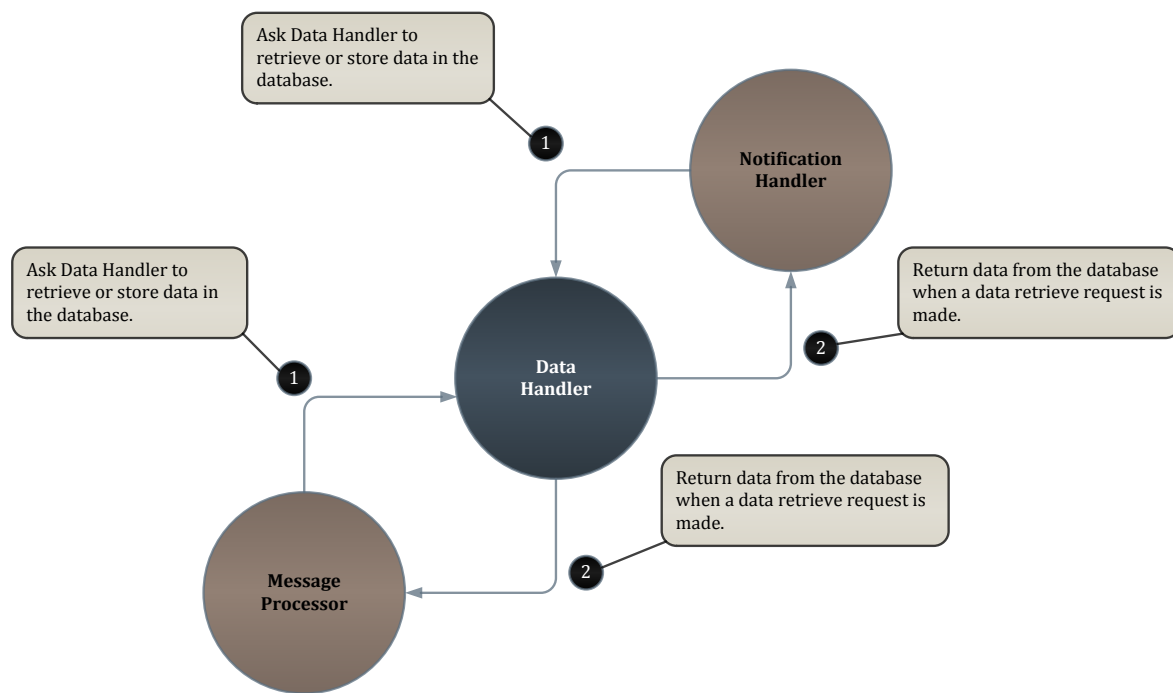
The following diagram presents the described flow:



Message Generator Data Flow

The **Data Handler** will be responsible for the interface between the CCBR system and the CCBR/ShipDB Shared Database. It will receive requests from the Message Processor and from the Notification Handler.

The following diagram presents the described flow:

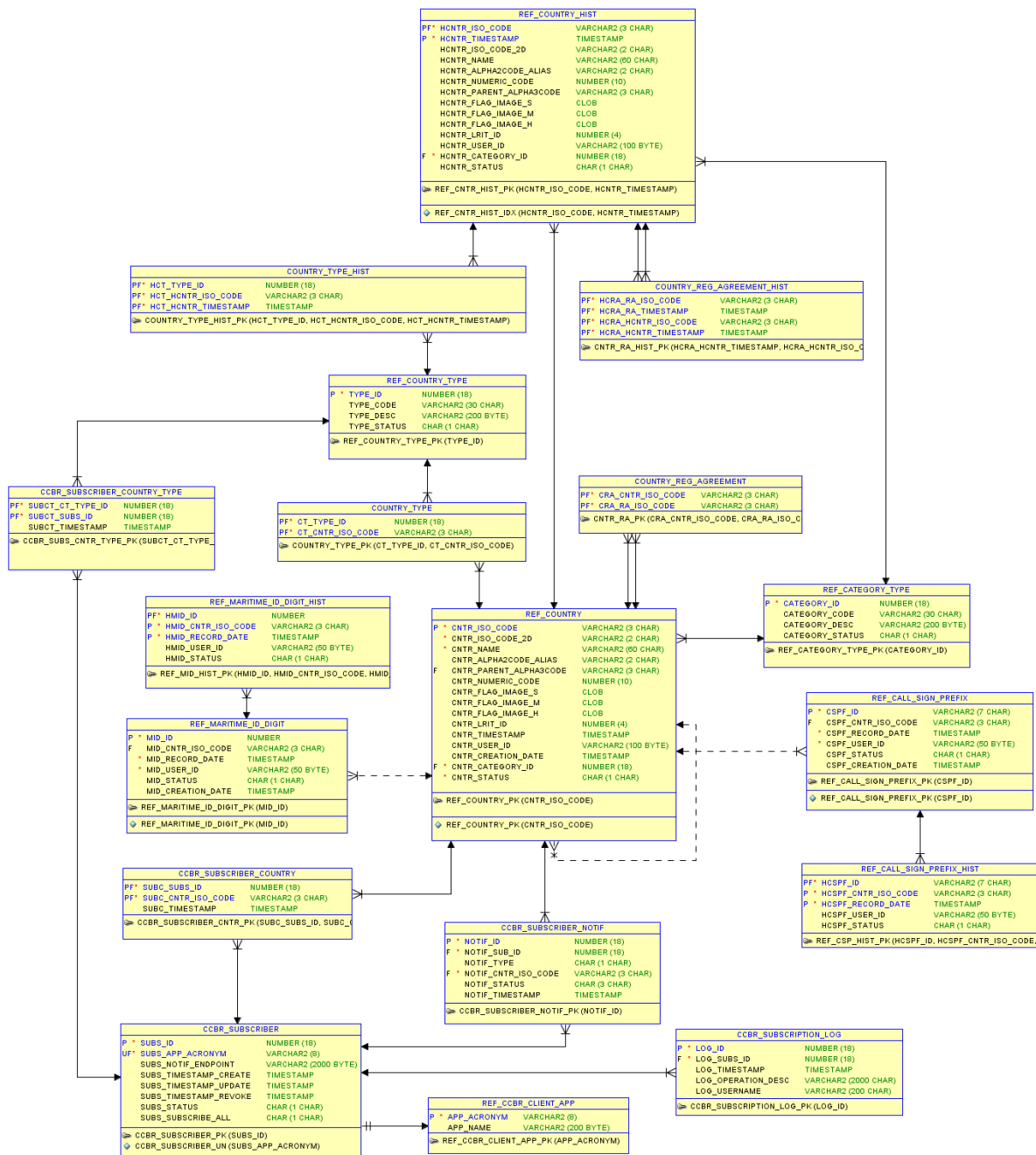


Data Handler Data Flow

4.2.1.4 CCBR/ShipDB Shared Database

The CCBR/ShipDB share database is a relational database handled by the Oracle RDBMS version 11g R2. The database access between the CCBR application (DAO's implemented by the Data Handler module) and the database will be performed using the JDBC driver via Hibernate.

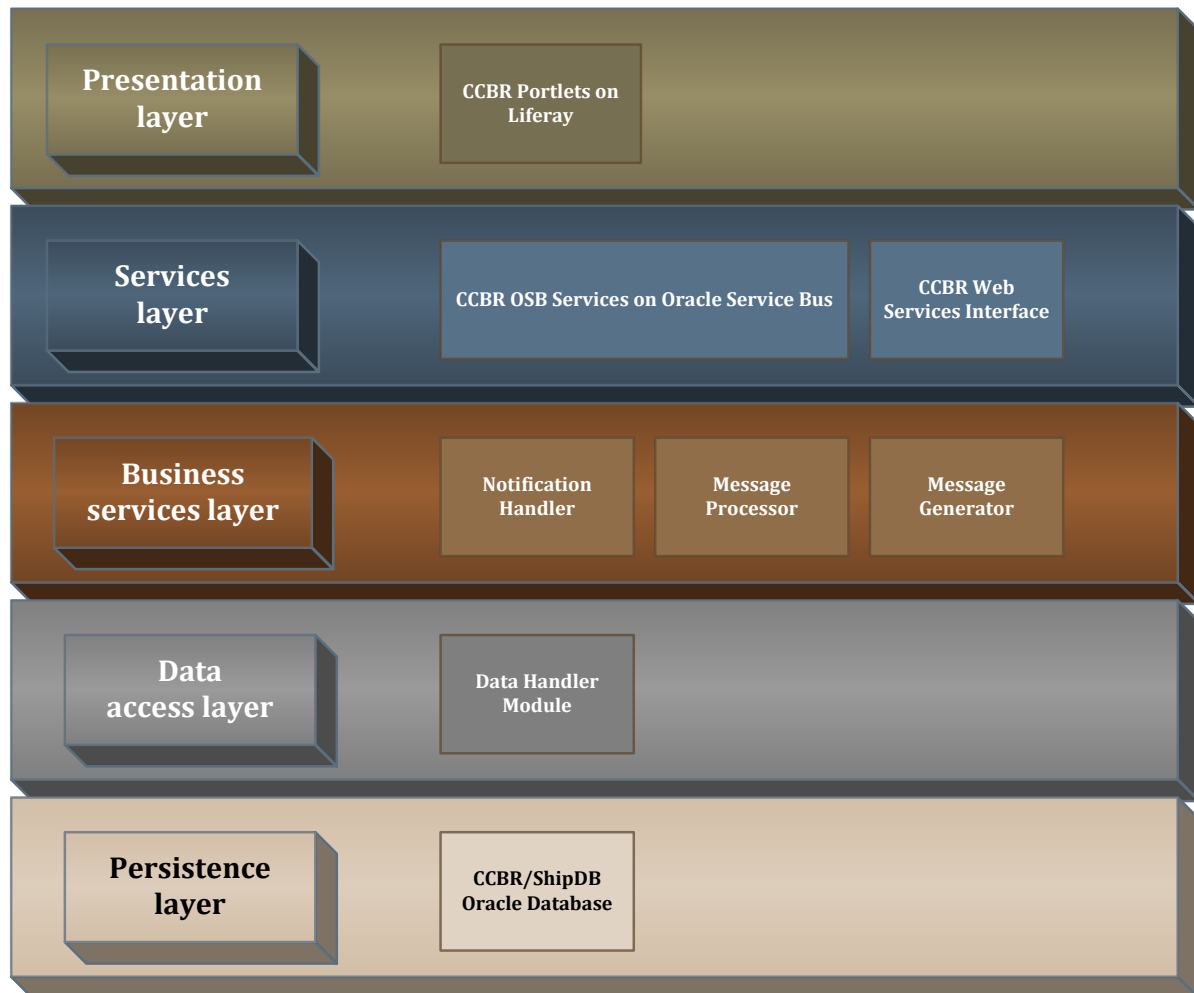
The figure below present the database model shared by the CCBR and ShipDB applications:



CCBR/ShipDB Shared Database Model

4.2.2 CCBR Layers Decomposition

This section follows a common approach in software engineering and, in particular, for J2EE applications: software layering. In this approach, each layer has its own functionality and the layers are organized in a way that each one provides support and base functionalities for other layers.



CCBR Layer Architecture

Best practices in OO programming organize software development into layers of responsibility: presentation layer, services layer, business layer, data access layer and persistent data layer.

The presentation layer interacts with users, typically through a Web browser. It is responsible for creating and displaying the user interface and handling user interaction. In the CCBR case, this layer will interface with the Service Layer in order to provide and collect data.

The Service Layer, also known as web service layer will be responsible for exposing the web service API to the clients. Through them, the CCBR clients will be able to access to CCBR functionalities and retrieve the needed information.

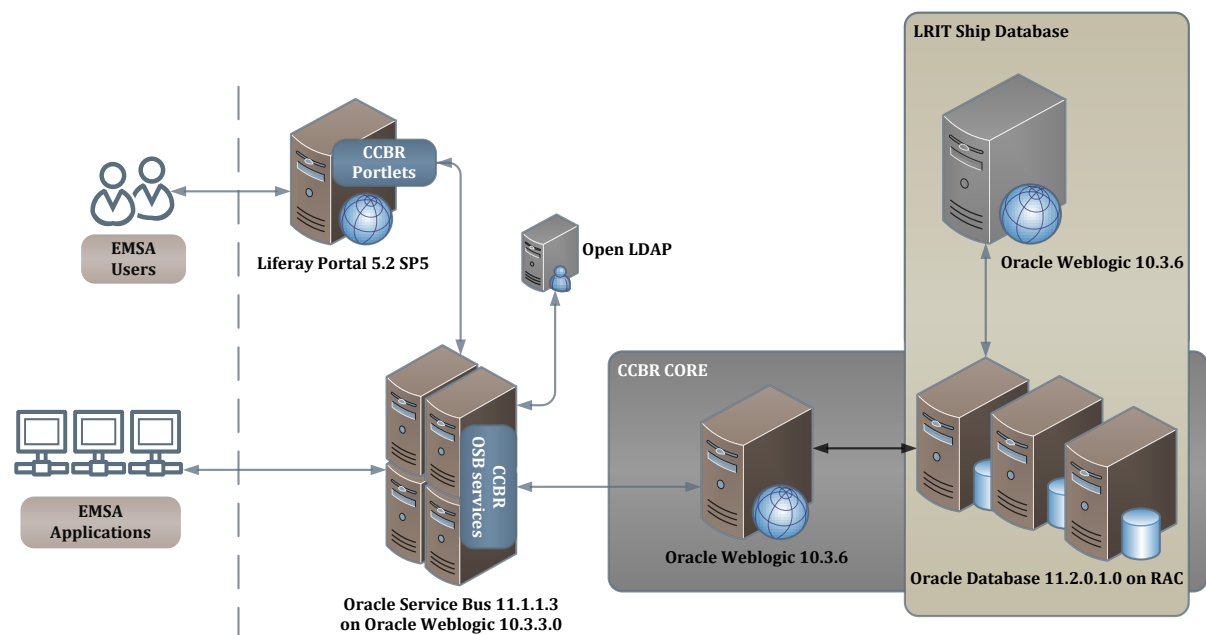
The Business Layer is responsible for all the business logic of the CCBR application. It's responsible for receiving requests from the upper layer, process them, and invoke the data access objects (DAOs) to return data in business objects. That information will be then sent to the Services layer in order to be delivered to the Presentation Layer or to the External Systems.

The Data Access Layer is responsible for querying the CCBR/ShipDB shared database in order to store information on it or retrieved stored information. It will receive requests from the Business Layer and interacts with the Persistence Layer to satisfy those requests.

Finally, the Persistence Layer, also known as persistent storage is where the CCBR information is stored. This layer is composed by an Oracle database.

4.2.3 CCBR Physical Diagram

The following figure presents the CCBR system physical diagram:



CCBR Physical Diagram

The user's access to the CCBR system is done through the web interface deployed in the EMSA Liferay Portal. By using this interface and according with the defined profiles, each user will have access to the country data and related information, subscribed data and performed operations.

The EMSA maritime applications access to the CCBR system is made through calls to the available web services deployed in the EMSA Oracle Service Bus Platform. The OSB will use the services of the EMSA Open LDAP service in order to validate the user's authentication and authorization.

The exchange of messages between the EMSA maritime applications and the CCBR system will be based on Simple Object Access Protocol (SOAP) and eXtensible Markup Language (XML).

The Core of the CCBR system will be running in the Weblogic Application Server.

The Oracle Database Servers running the Oracle 11g RAC (Real Application Cluster) are used to support data storage and data retrieval operations. The data itself is stored in the Oracle Storage Area Network (SAN). This architecture combination (RAC and SAN) provides high scalability and guarantees high availability and high performance, avoiding the disruption of the Oracle I/O activity.

At this moment is not foreseen the creation of a DR environment with the same architecture as the primary one to be activated in case of emergency (when the primary environment is down due to a natural or human-induced disaster) and to keep and guarantee the continuity of the CCBR service.

Since the communications between the users and the CCBR system is internal to EMSA, it's not foreseen, at this moment, the need of using SSL.

The Web Servers will be running using the following basic software:

- Red Hat Enterprise Linux (RHEL) OS Release 5.4
- Liferay Portal 5.2 SP5
- Oracle Service Bus 11.1.1.3 on Oracle Weblogic 10.3.3.0
- Oracle Weblogic 10.3.6
- Open LDAP

The Database Servers will be running using the following basic software:

- Red Hat Enterprise Linux (RHEL) OS Release 5.4
- Oracle Database 11g Release 2 (11.2.0.1.0) on Oracle Real Application Cluster (RAC) 11g