European Maritime Safety Agency

System and Application Technical Landscape

Version 36

Date: 18/05/2020

EMSA

Document History

Version	Date	Names	Comments
36	18/05/2020	EMSA	Published

Table of Contents

Acronyms	
AJAX	Asynchronous JavaScript and XML
BCF	Business Continuity Facility
BMP	Bean-Managed Persistence
CMP	Container-Managed Persistence
DAO	Data Access Object
DTO	Data Transfer Object
DB	Database
DC	Data Centre
DHTML	Dynamic HTML
DMZ	Demilitarized zone
DNS	Domain Name System
EIS	Enterprise Information System
EJB	Enterprise Java Bean
EMSA	European Maritime Safety Agency
ESB	Enterprise Service Bus
FTP	File Transfer Protocol
GIS	Geographic Information System
GUI	Graphical user interface
HA	High Availability
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol over Secure Socket Layer
IPSEC	Internet Protocol Security
ISP	Internet Service Provider
JCA	JAVA EE Connector Architecture
JDBC	Java Database Connectivity
JDK	Java Development Kit
JEE	Java Enterprise Edition
JMS	Java Message Service
JSF	Java Server Faces
JSP	Java Server Pages
JVM	Java Virtual Machine
LDAP	Lightweight Directory Access Protocol
Mbps	Megabit per second
MOM	Message Oriented Middleware
NAT	Network Address Translation
OAM	Oracle Access Management

OIM	Oracle Identitiy Management
OES	Oracle Entitlement Server
OS	Operating System
OSB	Oracle Service Bus
OWASP	Open Web Application Security Project
POJO	Plain Old Java Objects
R. Proxy	Reverse Proxy
RAC	Real Application Clusters
REST	Representational State Transfer
RIA	Rich Internet Applications
RMI	Remote Method of Invocation
SAN	Storage Area Network
SANS	SysAdmin, Audit, Network, Security Institute
sFTP	Secure File Transfer Protocol
SMTP	Simple Mail Transfer Protocol
SRM	Site Recovery Manager
SOA	Service Oriented Architecture
SSL	Secure Socket Layer
TB	Tera Bytes (i.e. 10^{12} bytes or 1 million mega bytes)
UDDI	Universal Description Discovery and Integration
VLAN	Virtual Local Area Network
VM	Virtual Machine
WLI	WebLogic Integrator
WLS	WebLogic Server
XHTML	Extensible Hypertext Mark-up Language
XWS	WS Security implementation from Sun Microsystems

List of Annexes	
Annex 1	“IAM Guide_abridged”
Annex 2	“EMSA secure development requirements v01”
Annex 3	“EMSA secure development recommendation guide v01”
Annex 4	“EMSA_JASPER_Technical_Document”
Annex 5	“EMSA SOA Guidelines & Rules”

1. Introduction and Objectives

This document describes EMSA System and Application landscape. Its main objective is to document the technical solutions used by EMSA at System level and to provide directions on options and preferable technologies to be considered at Application Level.

Although the System and Application Landscape described in this document are EMSA guiding lines, this does not mean that no deviations are allowed. Exceptions can be proposed and they will be considered on a case by case basis; if it is found that is the best technical implementation for the requirement or there is no other way of doing it, this exception will be accepted.
Also suggestions for innovation are welcome and if they bring added value to the landscape, they will be included.

The document is organized in several chapters:

- Chapter 1: Introduction and Objectives.
- Chapter 2: Describes the System Landscape and the Technical solutions implements at systems and network levels.
- Chapter 3: Describes the Application Landscape and preferable options to be used at the Application level.
- Chapter 4: Describes the conceptual Service Oriented Architecture (SOA) to which the applications should comply
- Chapter 5: Describes the software versioning scheme
- Chapter 6: A summary of the Software versions

2. System Landscape

2.1. High Level Network Schema

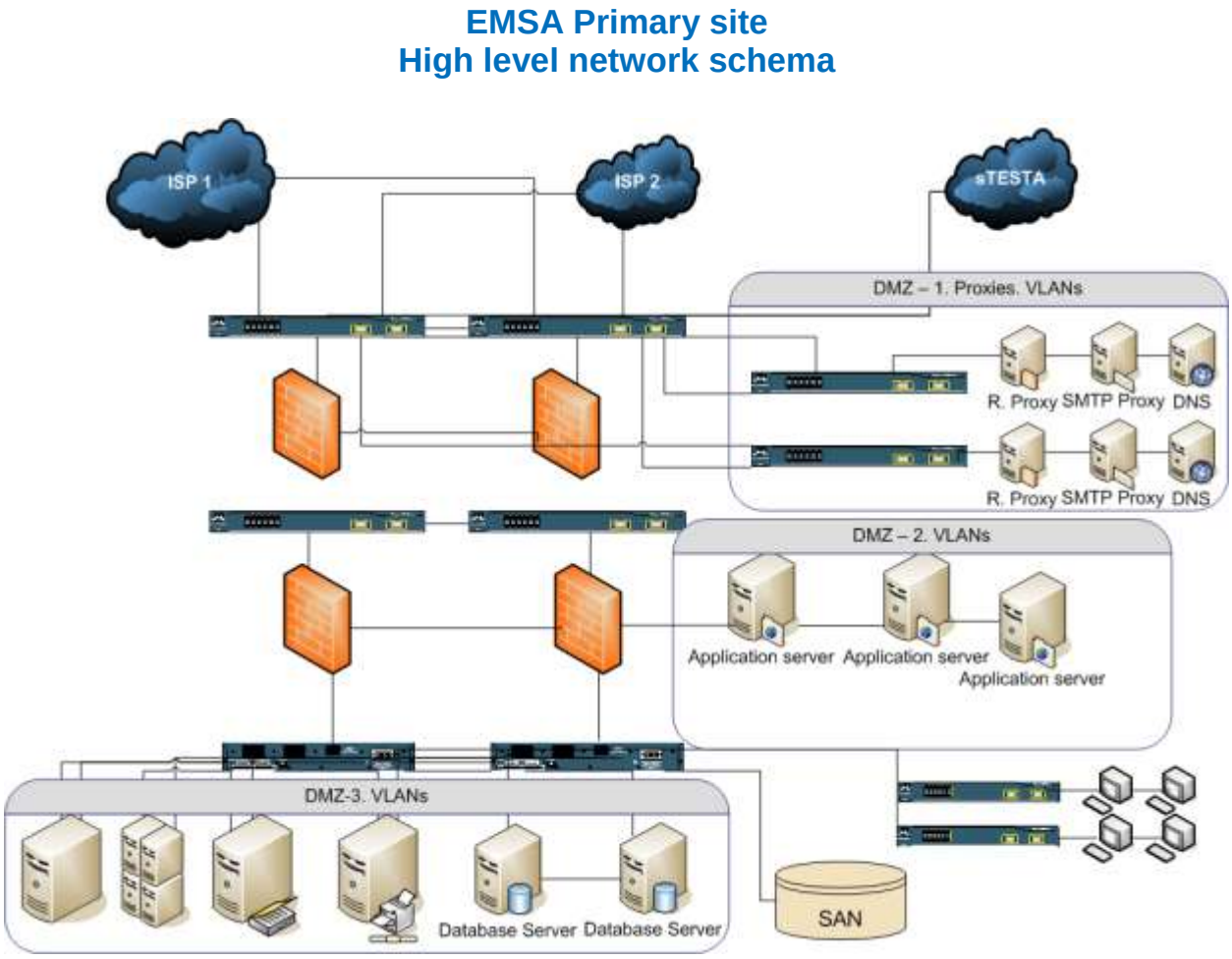


Figure 1 - Primary site. High level network schema

2.2. Data Links

Data Links
• 2 Internet ISP - active/active using BGP - BGP autonomous system and routing fully managed by EMSA - 100 Mbps each - 256 Provided independent IP addresses • 1 sTESTA link - EU private network - 2 Mbps • 1 GEANT link - Reserved to the CleanSeaNet project for high speed image transfer - 1 Gbps

2.3. Network Security

Two layers of firewall protection:

- Checkpoint R75.40 2-nodes clusters;
- Cisco ASA;

Reverse proxies for incoming connections (currently handling the following protocols: HTTP, HTTPS and SFTP). The network is segmented using VLAN's.

DMZs
•DMZ-1: reverse proxies, DNS servers, other services exposed to Internet
•DMZ-2: application servers and database servers (Front/Back End VLANs)

Monitoring of security events is currently achieved through a SIEM (Security Information Event Management) system including Suricata, Splunk, F5 ASM module on top of EMSA F5 reverse proxy.

2.4. Proxy Policy

The following rules should be followed:

- Accessing EMSA web applications should be always through HTTPS;
- Reverse proxies are used for all incoming connections from outside networks (Internet and sTESTA);
- All incoming connections shall pass through our reverse proxies;
- All incoming SSL connections are terminated in the reverse proxies;
- Proxies are always responsible for the SSL encryption and decryption;
- Proxies are always responsible for creation of the SSL connections;
- 1-way SSL is used for human to system interfaces while 2-way SSL should be used for system to system interfaces;
- All SSL outgoing connections shall use the proxy. Any outgoing SSL connection shall be initiated as plain HTTP by the applications to the proxy, where the SSL will be initiated for the outgoing SSL connection. The protocol used to request the proxy the creation of an outgoing HTTPS connection, involve the usage of an EMSA URL naming convention (<standard_URL>.f5 URL's) and some F5 configurations.

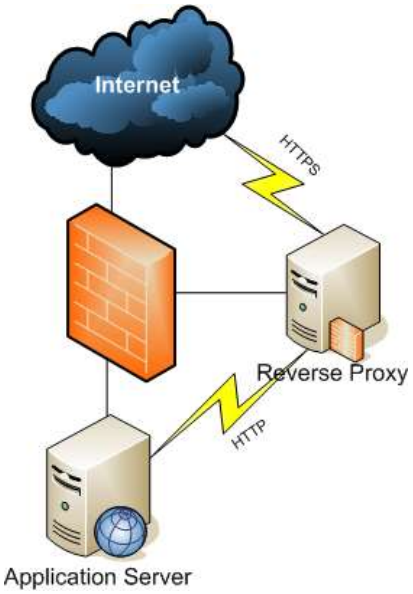


Figure 2: Proxy policy

Proxy Devices
• 2 x F5 Big IP v5000 Series

2.5. Network Load Balancing

The F5 appliances form a redundant cluster that can perform load balancing for web applications in any VLAN on EMSA network. The design of any new system or application should preferably implement load balancing with node fail detection on this equipment.

2.6. High Level Virtual Infrastructure Schema

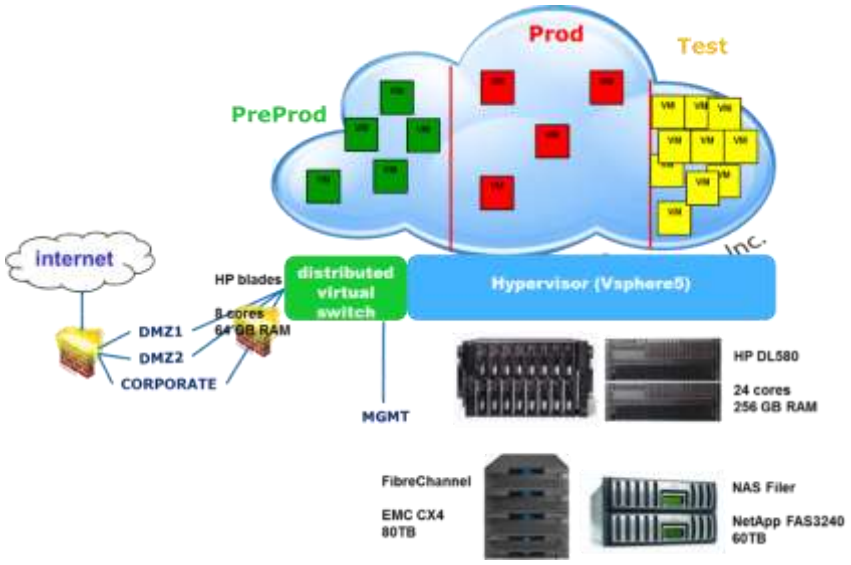


Figure 3 - High Level infrastructure

2.7. Virtual Infrastructure Services

The following services are offered to VMs and application environments:

- Basic monitoring with Nagios;
- Performance monitoring with vCenter Operations;
- VM-level backup with Networker or Netapp SnapMgr for Virtual Infrastructure. Exceptionally also Networker agent-based backup can be implemented.
- Deployment of a VM or environment[1];
- Cloning of a VM or environment;
- Snapshotting of a VM or environment[2];
- Exporting as OVF a VM or environment;
- Hardware resource allocation changes[3];
- Upgrade of VMware tools and virtual hardware;
- Troubleshooting.

2.8. Application Requirements For Virtual Infrastructure

Applications and systems hosted in the EMSA Virtual Datacentre must respect the following requirements:

- Base OS must be chosen out of the current EMSA template catalogue[4];
- Compatibility with the latest VMware virtual hardware specifications (currently version 8);
- Hardware provisioning done according to a principle of fit-for-purpose;
- Compatibility with vMotion.

2.9. Environments

EMSA has defined 6 possible different types of environments for the Maritime Applications. The following picture presents an overview of them.

[1] Subject to being included in the EMSA Template catalogue, currently including:
- Linux Red Hat Enterprise Server or CentOS in version 7;
- As above, with WebLogic or with Oracle DBMS;
- Latest Microsoft Windows servers.

[2] Subject to the following policy: the snapshot must be rolled back, or removed, in one week time to avoid performance penalties;
[3] Subject to the following policy: CPU, Memory, disk and network for any VM should be fit for purpose, and oversized VMs should be avoided to reduce contention issues and overhead. Granting more resources is subject to a trend analysis of the use of current resources also looking at vCenter Operations performance indicators, and takes into account its recommendation. VMs oversized are reported on a regular basis and are subject to downsizing.
[4] See note 1 on the previous page.

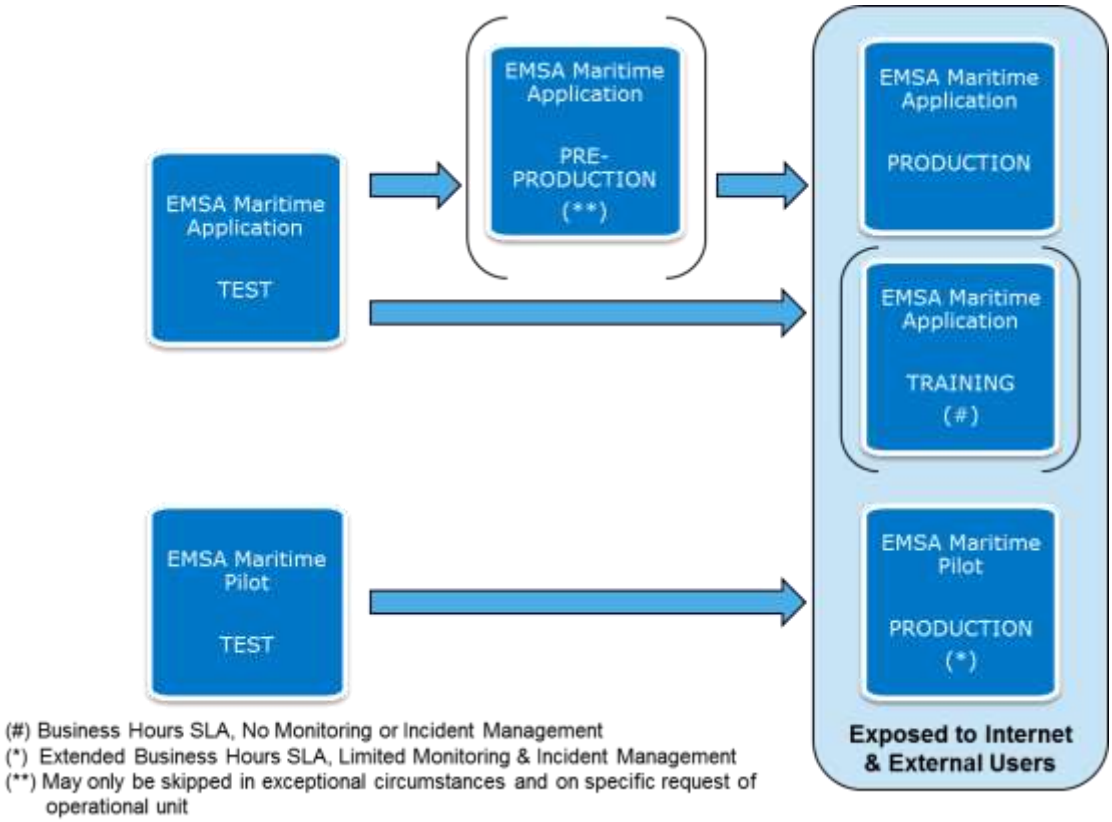


Figure 4: Types of Environments

The following figure shows detailed information related to each type of environment.

Type	InfraX grade	A.3 Monitoring	A.3 Incident Management	External access	Backup / Restore services (1)	Decommission Date	Ownership	VLANS
Test (t)	Test	No	No	No	Non Prod	No	Ops. Unit / Contractor - A.3 provides infraX + grant privileges - Ops.Unit handles it afterwards	Test
Pre-Production (pp)	Pre-Prod	No	Yes (Business Hours)	No or with IP Filtering	Non Prod	No	A.3	Pre-Prod
Training (i)	Pre-Prod	No	No	Yes	Non Prod	No	Ops. Unit - A.3 provides infraX + grant privileges - Ops.Unit handles it afterwards - A.3 act as support if needed	Pre-Prod
Production (p)	Prod	Yes (infraX and application outputs)	Yes (24 x 7)	Yes	Prod	No	A.3	Prod
Test Pilot (t)	Test	No	No	No	Non Prod	Yes (max. 1 year)	Ops. Unit / Contractor - A.3 provides infraX grant privileges - Ops.Unit handles it afterwards	Test
Pilot (i)	Prod (2)	Limited (max 5 checks per env, based on app outputs)	Yes (Extended Business Hours) MS5 – normal A3 - major	Yes	Non Prod	Yes (max. 1 year)	Ops. Unit - A.3 provides infraX + grant privileges - Ops.Unit handles it afterwards - A.3 act as support if needed	Pilot

(1) - Backup/Restore services:
- Non-Prod: VM: weekly + DB (RAC): constant / DB (non-RAC): weekly + File System (data): weekly
- Prod: VM: weekly + DB (RAC): constant + File System (data): daily

(2) – Production for Pilot Projects:
- only uses VMs with standard A.3 templates
- No clustering / load balancing
- Single instance DB (no RAC)

Figure 5: Characteristics per Type of Environments

The basic infrastructure that supports the environments is as follows:

Environments
• Production • Training: ideally 50% of the production capacity • Pilot Production: ideally 50% of the production capacity • Pre-Production: ideally 50% of the production capacity • Test/Quality: ideally 25% of the production capacity

Server Infrastructure
• EMSA Datacenter is fully virtualised with VMWare technologies • Those include: - VMware ESXi VSphere 5 - VMware HA, DRS and Failover

High availability technologies
• Service fail-over: Weblogic Active-Active, Oracle EXADATA • Server fail-over: VMware FailOver and VMware HA • Site fail-over: VMWare Site Recovery Manager; • Data replication: Asynchronous data replication via FCIP; backup storing off-site

Service Clustering
• Weblogic Active/Active clustering • Oracle EXADATA

SAN Storage
• Brocade fabric based on Sanswitch DS5300 • EMC Clariion CX4-240 • Netapp filer FAS3240 (only CIFS/NFSv3)

Critical applications and services must be mandatorily designed following High availability techniques (e.g. clustering) without any Single Point of Failure.

Environment	Test / Test Pilot	Pre-Production	Training	Pilot Production	Production
Purpose	This environment allows software contractors to perform testing and integration of their applications in the EMSA environment.	This environment offers a chance for EMSA application users to review and test applications in development or having past SAT.	This environment is used to perform training sessions with the end-users and MS commissioning tests.	This environment is used to implement new applications to validate new concepts before implementing a full-production system.	Shall only be provided for applications whose deliveries have been formally accepted. When an application is no longer in use, the application owner shall inform the ICT team of this change in status.

Infrastructure performance & scaling	Equivalent to 25% of production capacity	Equivalent to 50% of production capacity	Equivalent to 50% of production capacity	Equivalent to 50% of production capacity	
Responsibility and installation	In test environment the contractor will have the necessary privileges (limited to areas directly related to the development) in order to be able deploy the application under development without help from ICT teams. On request, ICT may make available staff to support the contractor.	The environment shall also be used to test installation procedures. Before any applications are installed or before configuration changes, data fixes, etc are performed, the contractor will deliver to EMSA all source code, installation scripts, installation procedures, release notes, etc, as described in the release management procedure. ICT will be responsible for installation and therefore the contractor or EMSA project officer will need to arrange with ICT, sufficiently beforehand, a date for installation.	In training environment the Operational Units will have the necessary privileges (limited to areas directly related to the development) in order to be able deploy the application under development without help from ICT staff. On request ICT may make available staff to support the contractor.	In Pilot Production environment the Operational Units will have the necessary privileges (limited to areas directly related to the development) in order to be able deploy the application under development without help from ICT staff. On request ICT may make available staff to support the contractor.	All software or scripts being run in the production environment shall first be installed in pre-production environment. Both EMSA business responsible and EMSA IT responsible shall have formally accepted the software in accordance with Software Release Management Procedure. Installation and maintenance will be performed solely by ICT or its contractors.

2.10. Disaster Recovery

EMSA's Business Continuity Facility (BCF) is hosted in the premises of a commercial hosting provider. The BCF is a fully equipped replica of the main site in terms of servers, network equipment, internet connectivity, storage and middleware, and as such it may function as either the main production site for an application, or as back-up site. This choice may be made on a per application basis and depends on the EMSA needs, the application's replication design and capabilities, and the desired SL.

Any new system or application must conform by design to one of the business continuity approaches foreseen so far:

1) **ON/OFF model**:

   The servers and services that constitute the system or application are active and visible on the network only in the main site. They are kept in sync in the secondary site with some middleware or low level replica technology like Dataguard for backends, or virtual machine cloning or storage array based replication for front ends. But the replicated systems are always inactive on the secondary site in an off-state and not visible on the network unless the recovery procedure is executed. Taking over in that case means executing a procedure to stop the systems in the main site (if possible), execute a last synchronisation (if possible), stop the synchronisation flows, then restart the replicated systems in the secondary site changing all the parameters that differ in the two sites like network configuration, internal DNS entries, pointers to database or cartographic servers or to any other horizontal service platform always available in both sites like LDAP, Single Sign On, DNS etc.… Eventually, the external DNS entry should be changed to point external Internet users to the public IP of the system or application in the new site.
   According to this model, it is still possible to have the same internal FQDN for the application servers in both sites, as servers are active and visible on the network only in one site at a time, and when taking over, the A records of the internal DNS can be changed to reflect the different IP address space in the new site.

2) **ON/ON model**:

   The servers and services that constitute the system or application are active and ready to take over at any time in both sites. Synchronisation rely on the features of the application or middleware used rather than on a low-level cloning and transferring of the virtual machines, offering either a fully multi-master active/active approach like Active Directory, or some type of distributed geo-cluster, or anyway an autonomous system which keeps data and configuration in sync between the two legs in the two sites.
   Taking over in that case is a simpler procedure like activating some built-in

system or application feature to switch to the other site, possibly requiring some internal and external DNS changes, or can be even fully transparent.

According to this model, different FQDNs and IPs for the application servers in the two sites must be chosen, as servers are active and visible on the network in both sites at any time.

Note: it is not accepted to design ON/ON systems where the virtual machines on the two sides have the same internal DNS FQDN.

The ON/ON model, when supported by the application or middleware, might guarantee faster and seamless fail-over procedure, hence it is the preferred approach.

The following figure exemplifies how the interconnection of current EMSA's production environment with the BCF is envisaged and points to the use of several replication/back-up systems at different levels of the infrastructure:

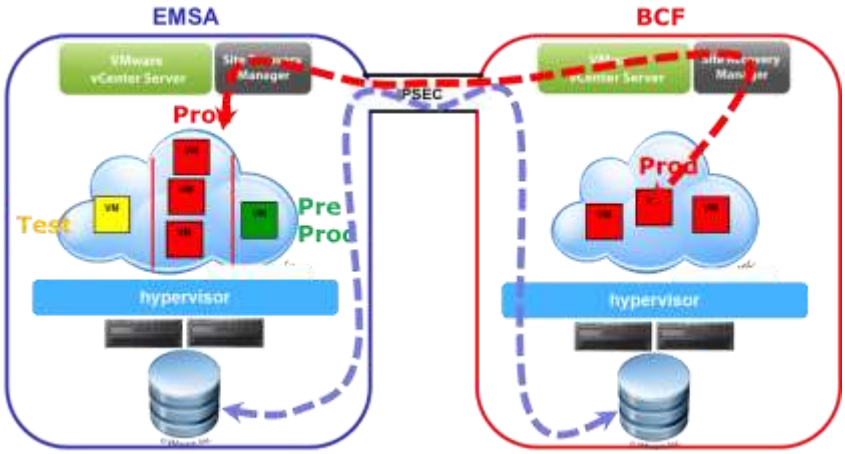


Figure 6: EMSA DC connection with BCF

Key elements of the actual BCF architecture are:

1) the two sites are connected through an IPSEC tunnel over an high performance link
2) the two sites use different private and public IP address ranges
3) the internal DNS zone emsa.local, containing server's FQDN, is shared between the two sites;
4) the external IP address space in each of the two sites is a different C-class of Provider Independent IPs whose routing advertisements is managed directly by EMSA routers
5) the external DNS zone "emsa.europa.eu" is unique across the sites, it is delegated to EMSA, and it is kept in sync between the two sites with master-slave DNS replication;
6) data and systems are kept in sync through either:
    a. Oracle Dataguard for backend;
    b. Storage array replication for most of the front end virtual machines;
    c. Ad hoc application built-in replication technologies, like active directory replication, or Microsoft continuous cluster replication for Exchange and SQL.
    d. Ad hoc scripts for data transfer.
7) Rerouting of Internet users to the BCF is done with DNS technologies

Applications development should always be *BCF friendly* by being compliant with the following requirements:

- Application shall never use IPs in any configuration or dependency.
- All relevant configurations must be externalized from the application; this can be achieved with properties files in the filesystem (never inside the application war, ear or deployment directory) or using a well identified table in the database.
- Application shall use FQDN in their configurations or references to any dependency.
- Bandwidth required for data and system alignment should be kept to a manageable amount to allow continuous replication over a non-dedicated medium bandwidth link. A bandwidth estimation for data synchronization between EMSA DC and BCF, through Oracle Data Guard and other technologies, shall be provided;
- A fail-over procedure to BCF shall be provided together with one to fail back to EMSA;
- A list of all the application configurations and dependencies which need to be resolved in the BCF and main production site for the application to run shall be provided:
    - Web services
    - Data sources
    - Other application(s)
    - Security constraints
    - Infrastructural services
    - Etc…

- Connections to other machines should always be configured by referring to the machine name, never by referring to the IP address directly.
- For critical system, BCF certification is mandatory

3. Application Landscape

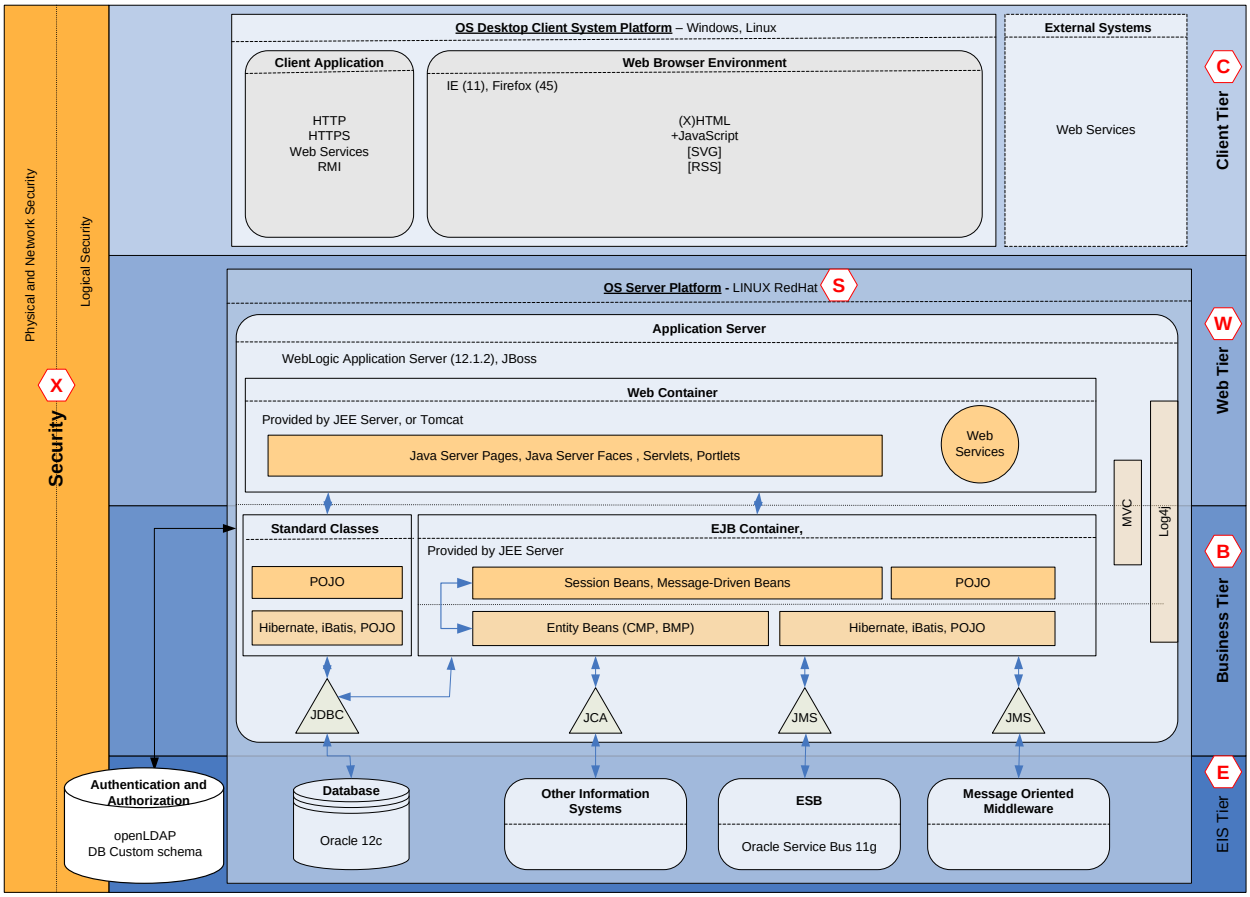


Figure 7: Application landscape

3.1. Architecture Overview

EMSA IT systems should follow state of the art JAVA PLATFORM, ENTERPRISE EDITION n-tier architecture. Figure 7 represents the preferable EMSA IT architecture where the major tiers are:

Client Environment
Client Tier: Client Tier is a JEE application front-end that provides communication with human users or with others external systems. For details, refer to chapter3.2

Server Environment
Web Tier: Web Tier connects user interface on a Client Tier with business logic on a Business Tier. For details, refer to chapter 3.3.1, (a)
Business Tier: Business Tier provides transaction processing logic (business logic) and data processing logic (data management). Business processes and business components should not be implemented outside this tier.

For details, refer to chapter 3.3.1, (b)
EIS Tier: EIS (Enterprise Information System) Tier consists of all enterprise information systems, such as databases or other information systems. ESB and Message Oriented Middleware are also included in this tier. For details, refer to chapter 3.3.2

Client Tier is the only tier of the Client Environment and it's by definition a distributed and separated tier.

Web Tier, Business Tier and EIS Tier are part of the Server Environment hosted at EMSA; EIS Tier (and its components) is usually a separated tier implemented on top of a separated server environment and depending on the complexity, the system architect may decide between a complete distributed architecture where all tiers are distributed in separated server environments or a mixed architecture where some tiers may share one server environment.

Operation systems options for the different environments are:

Client Environment
• Windows 10 • LINUX distribution desktop

Server Environment
• LINUX Redhat server 7 (64 bits) • Windows Server 2019

3.2. Client Environment and Client Tier

3.2.1. Web Browser Environment

The majority of EMSA applications are delivered to the final user via a browser based interface. A Web UI's advantage is that no additional software needs to be installed on client side and minimal demands are placed on the client platform.

Because a HTML Thin Client GUI is limited by markup language / JavaScript capabilities, others resources can add to build Rich Clients providing better user experience through the Web Browser. Applications must be 100% compatible with, at least, the following browsers or higher versions:

Web Browsers
• Microsoft Edge (latest versions) • Mozilla Firefox 70 and later

HTML page serves as a host for Rich Clients built with different technologies:

Client Tier Technologies
• HTML 5 • Plain Javascript and Tag Libraries • Single-Page Application, e.g. AngularJS, ExtJS,… (latest versions) • WebGL

Technologies used to implement Rich Internet Applications in the Client Tier can also have strong relationships with the technologies used in the Web Tier (e.g. Tag Libraries) described in chapter 3.3.1.

Usage of Java Applets should be limited to very particular situations and the decision to allow this will be taken on a case by case basis.

3.2.2. Client Application

Due to some business requirements (e.g. operation in disconnected mode, access to the local file system, …), some applications may require a Fat Client.

In order to create a unified technology platform, and to support all operating platforms in use at EMSA or EMSA clients, preference will be for using the Java language.

A mechanism for deploying and updating the client application at the remote PC will be needed (Java Webstart will be preferred). Dependencies on runtime components not already part of standard EMSA PC configurations will be regarded as negative.

Because EMSA needs to support other organisations within the Member States, any application to be installed on a client will need to be cross-platform, covering at least the platforms listed earlier in this document[5].
Usually, a client application will need also to connect to the server side of the system in order to perform business actions (e.g. data synchronization). Several technologies can be used to address this client-server connection; please refer to Annex 5 “EMSA SOA Guidelines & Rules” for details.

Communications to servers shall be done using web services, exceptions may be granted on request. Exposed Web Services shall always be protected with Authentication and Authorization. Important business data should always be stored on servers managed by ICT, if this requirement cannot be met (due to business requirements, impossibility to connect, …) a procedure for providing data back-ups needs to be foreseen.

[5] If the application is to be used only by EMSA this requirement can be reduced to supporting Windows 10. An application installer compatible with EMSA’s MS System Center needs to be provided.

In case development of a fat client is proposed, this needs to be discussed with ICT and agreements on installation requirements, connection technology and data back-up need to be reached before starting development.

Mobile application platforms
• iOS, latest versions
• Android, latest versions

Increasingly mobile devices are used for accessing web based information systems. Where possible, in order to avoid creating multiple platform dependent solutions, such developments should be based on simple website access, with appropriate changes applied to the UI to take into account the smaller screen size, reduced bandwidth and touch based controls used by mobile devices. In cases where business requirements cannot be reached using a mobile optimised website, at least the application platforms and version mentioned above need to be supported.

3.2.3. External Systems

External systems will also act as clients to EMSA systems creating the need of integrating different software systems used by different organizations (business partners). The system integration helps to automate collaboration processes and improve business performance. De-facto standard technologies should be used to inter-connect external systems with EMSA systems; please refer to Annex 5 “EMSA SOA Guidelines & Rules” for details.

3.3. Application Environment

3.3.1. Application Server

EMSA architecture is based on the standard JEE version 7. The following Application Servers should be used as the base Web and EJB containers:

Application Servers
• Weblogic Application Server (latest version)
• Wildfly/JBoss (latest version)

New development or ‘significant’[6] changes to existing applications should always target the latest version of the application server in use at EMSA. For existing applications, EMSA will assess the desirability vs the risks of upgrading the underlying application server on a case by case basis.

Simple applications, where distribution is not foreseen, the EJB container is not needed; see below for details.

[6] Significant shall be understood as any change resulting in a change of either major or minor versioning number (see further for a description of the version numbering scheme in use at EMSA)

(a) Web Tier

The delivery of Rich GUI based on Web Browsers is achieved by a set of components located in this tier and in close relationship with the Client Tier. Those components may vary depending on the technical solution adopted and level of complexity required for the Rich GUI; major technologies are presented in the next table:

Web Tier Technologies
• JSP – Java Server Pages
• JSF – Java Server Faces
• Portlets
• Rich server side components[7]

Portal technology
• Liferay Enterprise Edition

Simple applications, that only require a Web Container can use:

Web Container
• Tomcat (latest stable version)

Web Services are used to provide communication between loosely connected system components and are the preferable mechanism to expose services to external systems/applications. Several technologies could be adopted; please refer to Annex 5 “EMSA SOA Guidelines & Rules” for details.

(b) Business Tier

System functionalities are always implemented in the Business Tier and several technical options can be used to implement the Business components.
A software layer approach must be followed, implementing at least, two layers:

Business Layer: Responsible for the delivery of the business functionalities and orchestration of the business processes

Data Access Layer: Responsible for isolation of data access and actions executed over the persistent data storage (typically a relational database). Usually, Data Access Object (DAO) design pattern is mapped into this layer.

To support data transfer between layers and even between tiers a complete set of objects according to the Data Transfer Objects design pattern must be implemented.

[7] No preferable solution yet. On a case by case, other technologies that enable Rich Web base clients can be used

For simple applications where an EJB container is not required:

Business Layer technologies
• POJO (Plain Old Java Objects)

Data Access Layer technologies
• JPA • JDBC • Hibernate • springJDBC

For systems requiring an EJB container (that will be provided by the selected Application Server):

Business Layer technologies
• Session EJBs • Message Driven EJBs • POJO (Plain Old Java Objects)

Data Access Layer technologies
• Hibernate • springJDBC • Entity EJBs

3.3.2. EIS Tier

(a) Database

EMSA stores data in relational databases.

Relational Database Management System
• ORACLE 12c • PostgreSQL 12

New development or significant upgrades should enable the application to use the latest RDBMS version in use at EMSA.

(b) Message Oriented Middleware

To provide messaging services for integrated systems or asynchronous operations, EMSA relies on a Message-Oriented Middleware that increases the interoperability, portability, and flexibility by isolating the exposed services from the internal implementation and allowing distribution over multiple platforms (among other advantages).

Asynchronous messaging is the preferred method for exchanging data between internal applications. JMS will be the preferred manner for consuming and producing messages. The use of asynchronous message should enable better decoupling between applications (compared to web services), allow a more up-to-date system state (compared to batch processing), increased scalability (due to MOM underpinnings) and improved configurability and oversight of the system

integrations (through use of the ESB). Asynchronous messaging over JMS will also be the preferred method for request/reply messaging paradigm.

Message Oriented Middleware
• WebLogic JMS

(c) Other Information Systems

Any other Information Systems inside EMSA is considered to be in the EIS tier. Integration can be done using several techniques; preferable methods of integration are:

Internal systems integration technologies
• JCA – JAVA EE Connector Architecture
• Web Services; please refer to Annex 5 “EMSA SOA Guidelines & Rules” for details.

Asynchronous communication (based on call backs) should be used where possible.

Compared to the JMS based integration described above, more effort will be required to ensure the consumers / producers deal with service unavailability, scalability or reliability issues, therefore integration using asynchronous JMS is encouraged.

(d) Authentication and Authorization

EMSA owns a centralized system for Identity and Access Management; for details on this system, please refer to Annex 1, “IAM Guide_abridged”.

3.4. Security

Implementation of EMSA applications shall follow and be compliant with the best practices for secure programming. The standards detailed in Annex 2, “EMSA secure development requirements v01” are mandatory and recommendations described in Annex 3, “EMSA secure development recommendation guide v01” must always be taken into consideration

All applications shall be assessed against those recommendations and standards. These security assessments will be conducted by EMSA together with an independent external partner, at least once before entering PRODUCTION and whenever there is a EMSA’s decision to carried out a new assessment. Vulnerabilities found shall be addressed by the application implementing partner in agreement with EMSA.

3.5. Reporting Platform

Reporting Platform
• JasperReports
• Jasper BI

EMSA reporting platform is based on JASPER BI Enterprise Edittion; details on this platform can be found in Annex 4, “EMSA_JASPER_Technical_Document”.

3.6. Geographic Information System AND OGC (Open Geospatial Consortium) standards

EMSA Maritime Applications geospatial services are fully based on the OGC (Open Geospatial Consortium) standards, which have become key standards in use at EMSA. Some practical usage, but not limited, of these standards are:

- Electronic Nautical Charts:

EMSA is currently using an Electronic Nautical Charts distribution system for usage on the EMSA Maritime Applications. This system is providing ENCs, using a standard WMS interface, that are used as the base layer on the EMSA Maritime Applications.

- OGC standards used for vessel detection and correlation

EMSA is using OGC standards to provide Vessel Detection and Correlation services to other EU agencies. The standards being used are WMS and WFS. It is also intended to use WPS to generate VDS (Vessel Detection System) correlations.

- Creation of traffic density maps

EMSA is now using OGC standards (mainly WMS) to provide traffic density maps to end-users. EMSA will develop further this functionality to include more detailed TDMs with a higher definition than the current maps, on smaller areas), comparative maps (which show the differences between two maps) and vector maps (which show individual ship routes in polylines).

The GIS technologies in use at EMSA are:

GIS Platform
• ESRI Arc GIS • Jeppesen C-Map Professional + • GeoServer • Luciad

3.7. Logging

Log4J shall be the preferred library for generating application logs. All application logs should use the same log message format, as described below:

```
<param name="ConversionPattern" value="%d{yyyy-MM-
dd/HH:mm:ss.SSS/zzz} %-5p [%-t] [%l] %x - %m%n" />
```

Mandatory fields and format:

- `%d` – date in the specified format

- `%-5p` - Priority of the logging event.
- `%m` - application supplied message associated with the logging event.
- `%-t` - name of the thread that generated the logging event.
- `%l` - location information of the caller which generated the logging event.
- `%x` - NDC (nested diagnostic context) associated with the thread that generated the logging event.

The following conversion patterns should be avoided as much as possible for Production environments, due to increased processing needs:

- `C`
- `F`
- `1, L`
- `M`

The logging level should be changeable without requiring a restart of either the application or the application server. As for all configuration files, the log configuration file must reside outside of the packaged application.

Definition and implementation of log rotation and clean-up rules/processes is mandatory for every single logfile generated by the systems and its components.

EMSA makes use of Splunk for logging centralization and visualization. Applications must make sure that the logging patterns used are compatible with Splunk.

3.8. Storing Times and Dates

All EMSA servers, regardless of their function, shall use NTP to maintain accurate and aligned system clocks.

In order to prevent mismatches between data stored in different applications, all data shall in all cases be stored in Coordinated Universal Time (UTC). It is important to note that UTC, as opposed to local time, does not change with a change of seasons.

When a time is displayed to a user, used for triggering workflows or generating reports, it shall be the responsibility of the application to convert, if so desired, the stored UTC time to local time for the user. The final decision on if, or how the conversion shall happen, depends on the business requirements and will be an application decision. It is recommended for the user to be informed whether UTC time, user local time or source local time is displayed.

3.9. Others

The following points are generic mandatory requirements that shall be respected:

- Root or rooted administration accounts shall not be used.

- All system components shall be used by the same OS user.
- Software distribution cannot be done using rpm or any other solution that requires root privileges.
- In case it is necessary to have authentication on middleware components (e.g. application server, JMS) a dedicated user must be used. This user cannot be administration user of the components.
- When using non-compiled languages (e.g. php, perl) the versions of these languages shall be aligned with the version distributed bundled in OS version.
- Configuration files shall not include passwords in clear text. Solution to cope with this requirement may vary and must be agreed with EMSA.

If any deviation is foreseen, it shall be detailed and justified. EMSA has the last word in the decision process.

4. Service Oriented Architecture

EMSA applications should be compliant with the Enterprise Service Oriented Architecture with the objective of providing business and data services to others applications and being flexible and agile in order to easily adapt to change in short time.

EMSA Service Oriented Architecture is supported by a state of the art Service Oriented Infrastructure that follows the architectural best practices of the SOA metamodel.

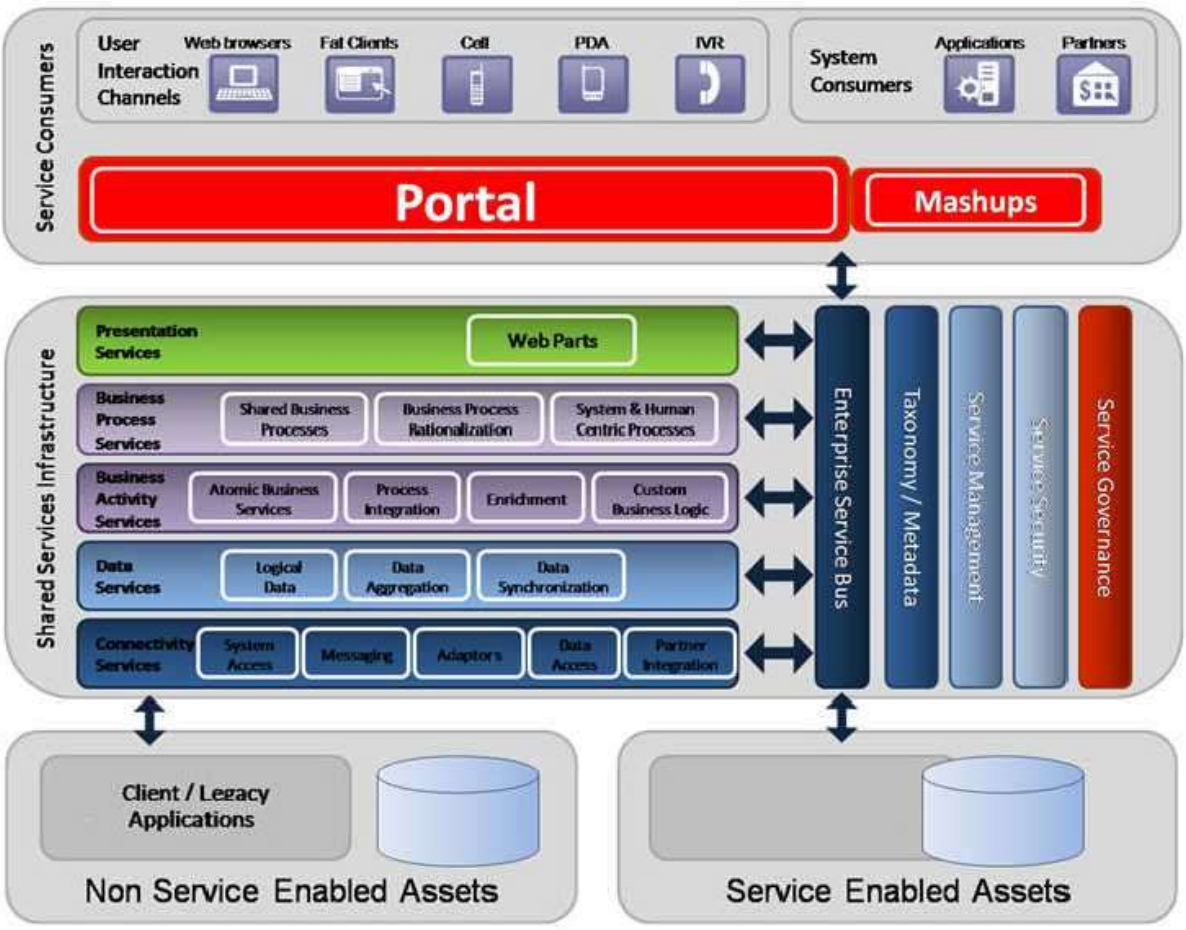


Figure 8: SOA architecture

The two major components supporting EMSA Service Oriented Architecture are:

EMSA SOA key components
• Liferay Portal, version 7.1 Enterprise Edition • Oracle SOA Suite 12c

The fundamental building block of Service Oriented Architecture is a service. A service is a component that can be interacted with through well-defined interfaces or message exchanges. Services must be designed to perform simple, granular functions with limited knowledge of how messages are passed to or retrieved from and for flexibility, agility, availability and stability.

EMSA principles of service orientation, which must be followed while designing services, are:

1. Services are loosely coupled components
2. Services are independent components
3. Services are self-contained
4. Services boundaries are explicit
5. Services are autonomous
6. Services share schema and contract
7. Services are independent deployable (logical aggregation can be considered)

Services designed based on these principles are much more likely to be reused within EMSA growing SOA infrastructure.

Please refer also to Annex 5 “EMSA SOA Guidelines & Rules”.

4.1. Service Consumers

Service consumers or composite applications are the applications that are developed to handle business actions or events initiated by business initiators. Business event initiators are entities that initiate business actions or events (either human users or other systems).

4.2. Shared Service Infrastructure

Shared service infrastructure defines the framework to shared services. It is based on Validate, Enrich, Transform, Route, and Operate or invokes (VETRO) patterns

Shared services are shared and reusable services that are used in service orchestration while creating business processes. Examples of shared services types are:

- Presentation services that present the data to the user.
- Business services that represent core business capabilities. Business services can range from relatively simple to very complex cross-functional, inter-enterprise business process.
- Data services that are entity services which provide access to enterprise data. Simple data services have a Validate, Create, Retrieve, Update, and Delete (CRUD) interface but more complex data services could be responsible for data aggregation or data synchronization.

5. Software Versioning Scheme

All applications being developed for or by EMSA shall use the following versioning scheme:

- [major].[minor].[revision]<.internal number>

Follows a description of the fields:

- Major will start 0 and will be increased by 1 every time significant new functionality is added to the application, or when significant changes to the implementation and/or organisation of the code have happened, such as:
  - When delivery of a new application or a major new version has been accepted, the major number will be increased by 1, other version numbers will be reset to 0;
  - Development of the next major version starts by increasing major version number by 1 and resetting all other version numbers to 0;
  - The above rules mean that all even numbered versions (+0) will be development releases for major new versions, whilst all odd numbered versions will be stable, production releases. E.g. if a software with version number 0.2.65 has been accepted for use in production environment, its version number will be 1.0.0. Development for the next major release will start at 2.0.0 and the production accepted release of this will carry a 3.0.0 version;
- Minor will be increased by 1 whenever less important new functionality or user interface changes are introduced;
- Revision will be increased by 1 whenever a new application version containing only bug fixes is delivered for deployment in EMSA pre-production environment;
- The internal number is an optional element that may be used by the contractor.

6. Summary

Minimum SW versions

SW or Technology	SW Version	Comment
Oracle WebLogic	12.2.1	Active / Active Weblogic clustering is foreseen for critical applications
Wildfly	10.1	
Tomcat	9.0	
Oracle IdM Suite	ORACLE IAM Suite 11gR3 IAM 11.1.2.3.0 SOA 11.1.1.9.0	
Oracle Access Manager	OAM suite 10gR3 OAM 10.1.4.3.0	
Oracle SOA Suite	12.2.1	
Oracle OSB	12.2.1	
openLDAP	2.4	
Jasper BI Jasper Reports	7.1	
Liferay	DXP 7.1	
ORACLE EXADATA database	12.1.0.2	
ORACLE standalone database	12.1.0.2	TEST environment only
PostgresQL	12	
ESRI ARCGIS	10	
Geoserver	2.14	
LuciadLightspeed	2016.1.53	
LuciadFusion	2016.1.53	
LuciadRia	2017.1	

Please note that, based on EMSA's official patching policies, the mentioned versions can be changed in specific cycles and without notice. Therefore, the above versions shall be considered as the minimum versions and never as "the only version"

Some additional information, can also be found below:

Area	Description	Technology	SW Version	Comment
Backup	SW	VMware VM backup; Legato Networker	7.6 SP3	
	HW	HP MSL8096 and Dell PVT Tape Libraries	N/A	
Business Continuity	HW/SW systems to guarantee different degrees of service availability	Local scale: VMware HA and FailOver Geographial scale: Asynchronous data replication through the Storage Array;	ESXi V 5	

		VMWare Site Recovery Manager;		
Clustering	Service fail-over	Front-end: Weblogic Active/Active Back-end: Oracle EXADATA	12c 12c	
Data Links	Internet connectivity	2 Internet circuits Internet IP connections	N/A	Each link: 100 Mbps, 256 Provided independent IP addresses
GIS		ESRI ArcGIS Geoserver,	10 2.14	
HW Servers	VM hardware	VMware Hardware revision 8 (vSphere 5)		Only production database is not virtualised and runs on blades as well.
	VM Host hardware	HP Blade and DL series servers	N/A	
Monitoring System		Nagios	N/A	
Network Security	Security DMZ	Checkpoint blades	R75.40	2 node clustered configuration with Mobile Access VPN
Operating Systems		Linux and MS Windows	RedHat Enterprise Linux 7 Windows Server 2008	
Proxy	Security DMZ	F5 Big IP v5000 series proxies	11.4.0	Clustered configuration with 2 nodes
SAN Storage	Storage Area Network	Brocade Fabric; EMC Clariion Model CX4-240; Netapp FAS3240		
Virtualisation		VMWare	vSphere 5	
Electronica Nautical Charts		Jeppesen C-Map Professional +	V360	For redundancy purposes: 2 nodes load-balanced in the F5

European Maritime Safety Agency

IAM Guide

Identity and Access Management Guide

(Abridged Version)

Document History

Title	**Identity and Access Management Guide** (abridged version)
Version	2.3 from 18/05/2020

Table of Contents

Table of Figures

Definitions, acronyms and abbreviations

Definition	Description
AccMng	Access Management
AD	Microsoft Active Directory
BCF	Business Continuity Framework
CMC	Common Management Console
Country/Institution	Defines the Nationality of a User and the area of control of a National Administrator.
CSN2	Clean Sea Net 2 Maritime application – version 2
EMSA	European Maritime Safety Agency
IAM	Identity and Access Management
IdM	Identity Management which comprises Access and User Identity Management
IdM V2	Identity and Access Management, version 2
IMDatE	Integrated Maritime Data Environment Maritime application
JAAS	Java Authentication and Authorization Service
LDAP	Lightweight Directory Access Protocol
LRITDC	Long-Range Identification and Tracking Data Centre Maritime application
JSON	JavaScript Object Notation. Lightweight data-interchange format
MAP	Maritime Application Portal (Liferay customisation of entry page to act as an “access point” for all of EMSA’s Maritime Applications)
MarApps	Abbreviated form of referring to EMSA Maritime Applications
MSS	EMSA’s Maritime Support Services
OAM	Oracle Access Management
OIM	Oracle Identity Management
Operation	Defines an Action that is available to a User.
Organization	Defines the Organization a User belongs to and the area of control of a Local Administrator.
OSB	Oracle Service Bus
OVD	Oracle Virtual Directory
RAC	Oracle Real time Application Cluster
REST	Representational State Transfer. Web Services that conform to the REST architectural style
RuleCheck	Application providing EU and International legislation regarding Port State Control
SAP	Webgate Specific Access Point configuration
SEG	SafeSeaNet Eco-system GUI
Service	Represents a set of (one or more) Business Functions implemented by an application (MarApp).
SOA	Service Oriented Architecture
SSN	Safe Sea Net Maritime application
SSO	Single Sign-On
STCW	Standards of Training Certification and Watchkeeping Maritime application
THETIS	The Hybrid European Targeting and Inspection System Maritime application
UMC	User Management Console
WebGate	Secured access entry point for applications

1. Introduction and objectives

This document describes EMSA Access and Identity Management. Its main purpose is to document the technical solutions used by EMSA to implement Access Control and User Identity Management throughout EMSA systems and applications.

It should be noted that this is an abridged version of the original document intended only for obtaining a high level perception of EMSA Access and Identity Management.

During the past years, EMSA has developed a common infrastructure to provide Identity and Access Management (IAM) services to the EMSA Maritime Applications.

IAM suggests that each user assume a unique digital identity across applications and systems, which enables access control to be assigned and evaluated against this identity at a central place as well as centralized management of user attributes. Thus, the IAM concept encompasses two major areas:

- **Access Management** managing authentication and authorization to resources and Single Sign-On (SSO) which is a mechanism whereby a single action of user authentication and authorization can permit a user to access all applications and systems where he has access permission, without the need to enter multiple passwords.

  Currently at EMSA, **Oracle Access Manager** (**OAM**) 10gR3 (10.1.4.3.0) is used to provide base Access Management and Single Sign-on functionalities.

- **Identity Management** is the management of the unique digital identity, associated attributes, security model and permission behind it. The set of user attributes varies from application to application and includes, among others, First Name, Last Name, Email, Groups and Roles. The security model establishes the management relationships (e.g. who is entitled to create/edit other users) and the permission rules (e.g. a Service Administrator can create users inside his own Service (application) and a National Service Administrator can create users belonging only to his own country for the service he manages). In addition, Identity Management also provides user's attributes and roles assignments to all applications that the user has access through a background provisioning process or through dedicated services.

  Currently at EMSA, **Oracle Identity Manager** (**OIM**) 11gR2 is used to provide base Identity Management functionalities.

The IAM service conveys benefits to an enterprise through:

- Central user repository for all applications and systems;
- Central User Management avoiding different implementations and rules across the enterprise;
- Reduction of human errors, a major component of systems failure, therefore highly desirable but difficult to implement;
- Reduction in the time taken by users in sign-on operations to individual domains, including reducing the possibility of such sign-on operations failing;
- Improved security due to the reduced need for a user to handle and remember multiple sets of authentication information;

- Reduction in the time taken, and improved response, by system administrators in adding and removing users to the system or modifying their access rights;
- Improved security through the enhanced ability of system administrators to maintain the integrity of user account configuration including the ability to inhibit or remove an individual user's access to all system resources in a coordinated and consistent manner;
- Significantly reduce the User Management maintenance and operation effort.

The document is organized in several chapters:

- Chapter 1: Introduction and objectives. This chapter;
- Chapter 2: EMSA IAM technical overview. A quick description of the architecture and components support EMSA IAM.
- Chapter 3: Access Management. Focus is given to the principles and implementation of EMSA's Access Management infrastructure.
- Chapter 4: Identity Management. Focus is given to the principles and implementation of EMSA's Identity Management infrastructure.

One final note about this document, as it is intended to be a guide used for presenting the information on reasons, implementations, etc. it is not necessarily supposed to be read "as a book", i.e. from the beginning to the end in a sequential manner. This document is more of a look-up to certain details and consequently may repeat information or "state the obvious" in some parts which have already been spoken about in other parts or in other documents.

2. EMSA IAM technical overview

The Introduction and objectives chapter presented the concept of IAM as implemented at EMSA. The next figure shows the context diagram of EMSA's IAM framework:

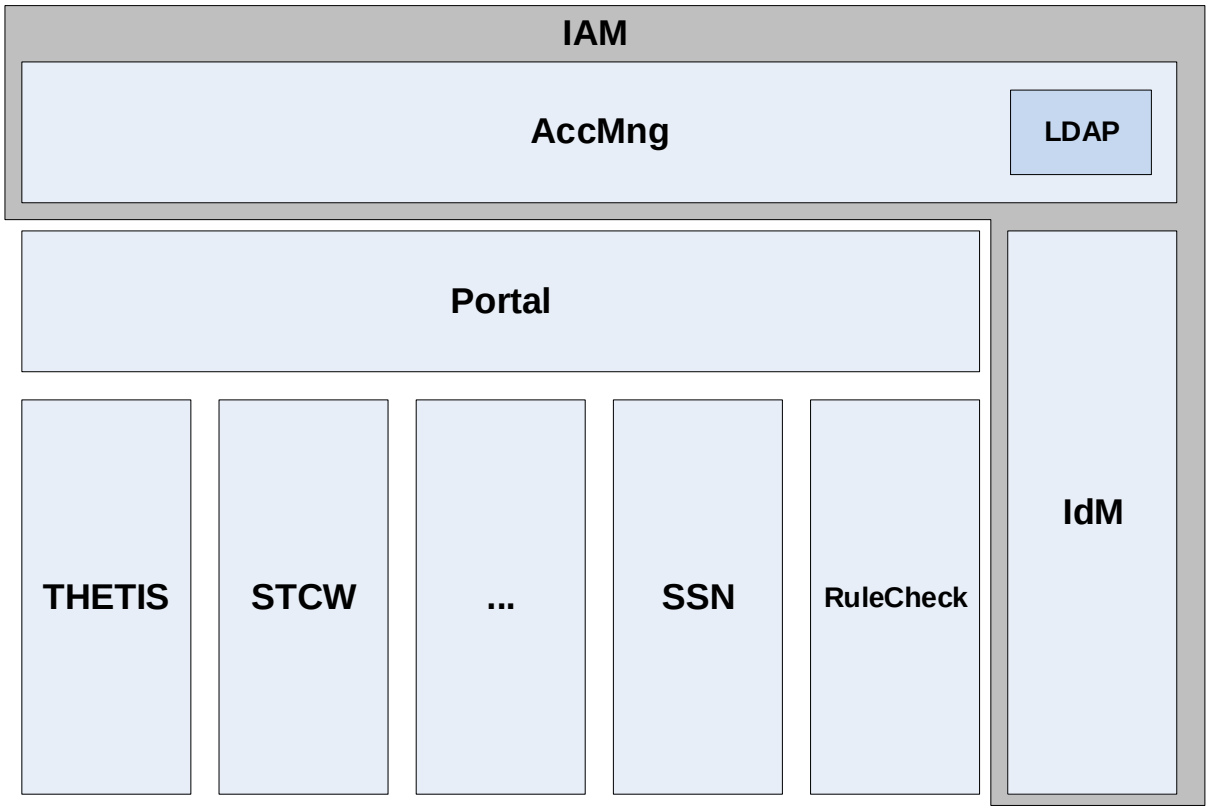


Figure 1: Context Diagram

The EMSA IAM framework provides governance for the accesses to EMSA applications. In a very simple way, we can say that:

- AccMng (Access Management) grants access to EMSA applications, providing Single Sign-On capabilities across those applications;
- IdM (Identity Management) manages the accounts (entities) that are entitled to use EMSA applications, providing functionalities like creation of new accounts or modification of existent accounts (changes of account attributes or Roles). It should be noted that AccMng also provides SSO functionality to IdM;
- LDAP is the central repository for access management maintaining information on accounts, roles and associations between accounts and their assigned roles;
- EMSA's Portal solution is built upon Liferay Portal. The Portal provides a single point of entry for several EMSA Maritime Applications (STCW-IS, THETIS, …) and, for those applications, also takes care of the authentication process by interacting with AccMng;
- EMSA applications (STCW-IS, THETIS, …), commonly referred to as Maritime Applications (MarApps), implement and provide EMSA core business functionalities.

The following diagram depicts the same information as the previous diagram, providing a deeper view of the different technical components used and includes the basic flow of requests. However, the machines depicted are purely "logical" and may not correspond to

actual physical machines (these may be single, clustered or joined together depending on actual implementation constraints).

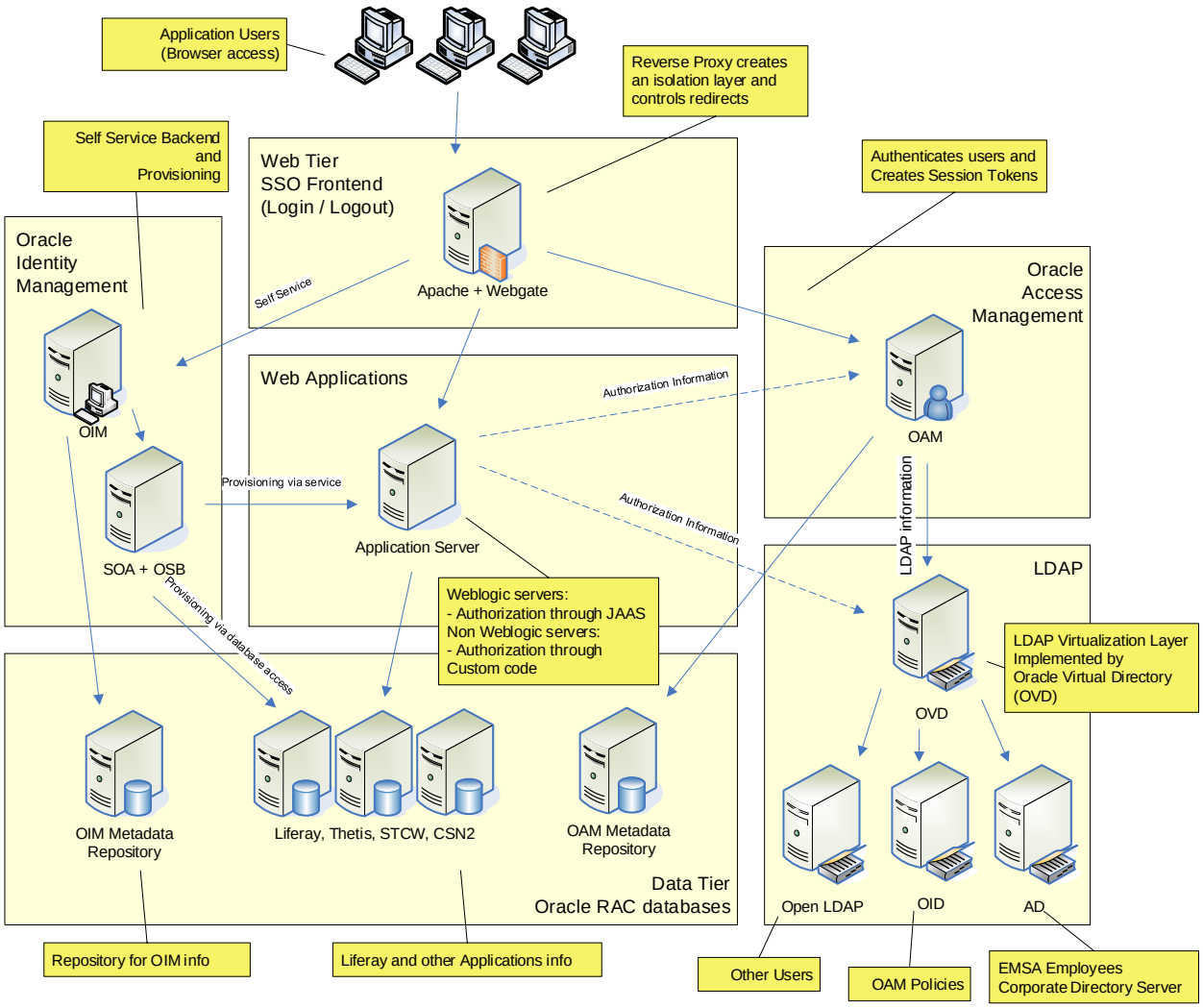


Figure 2: Technical Components

The components of the IAM high level blocks depicted above are identified below:

- AccMng, Access Management, is composed of:
  - SSO Frontend (Apache + Webgate) + OAM + LDAP virtualization
  - Data repository
- IdM, Identity Management, is composed of:
  - OIM
  - SOA Suite + OSB
  - Data repository
- Web Applications
  - Please note that this block aggregates Portal and Maritime Applications (THETIS, STCW, SSN, ….)

From the Access Management point of view, in this diagram it is possible to see that all accesses are made through the Apache Server and Webgate module (acting as a reverse

proxy). From here, if users are already authenticated, they may be permitted to access the web applications[1] (Apache + Webgate -> Web Applications). If the users are not yet authenticated, they will be shown a Login Form from OAM for authenticating (Apache + Webgate -> OAM). After the users submit their credentials, these will be verified by OAM on the LDAP virtualization layer[2] (OAM -> OVD) and if they are correct, a Session Token will be generated and returned to Apache for inclusion in all subsequent requests. Apache then redirects the user to the original URL requested. This authentication mechanism is used for all accesses that go through the Apache reverse proxy.
If the URL requested is part of the OIM self-service (Apache + Webgate -> OIM), there is a guarantee that users have already been authenticated and the corresponding functionality will be accessed. Depending on the action requested, OIM may do provisioning work through a service interface (OIM -> SOA+OSB -> Web Application) or just store information inside its own database to be accessed through specific Web Services.
If the URL requested corresponded to a web application (Apache + Webgate -> Web Applications), then the respective application may request Authorization information from OAM (Web Applications -> OAM). The exact process through which this is done will depend upon the application servers used.
If WebLogic is used, a JAAS integration might be best option; if not, a call to the OAM API through custom code will need to be done.

Note that, although not represented in the diagram (for clarity reasons), LDAP is usually provisioned (OIM -> SOA+OSB -> openLDAP) with the accounts information to serve as the base for the Authentication and Authorization process described above.

[1] "Web Applications" refers to Portal, THETIS, STCW, ….
[2] Although shown in the diagram, corporate AD is not integrated

3. Access Management

3.1. PROTECTING APPLICATIONS

EMSA hosts several Maritime Applications (MarApps), most of which deal with sensitive information that needs to be protected and or restricted. To reach this goal the MarApps have a series of protective layers:

- The first layer of protection is provided through the IdM Single Sign-On (SSO) mechanism which only allows access to pre-identified persons.
- A second layer could be implemented through the OAM Access Policies only allowing access to specific URL's when users belong to specific LDAP groups.
- Any layers from this point onward can be considered as application dependent and must be implemented inside the respective applications (i.e. application roles and/or specific business functionality access permissions).

This document only considers the first two layers leaving the other layers to each individual MarApp. It is worth mentioning that the second layer is not currently used to its full potential.

3.2. AUTHENTICATION

The general concept of Authentication can be defined as "the process of determining whether someone or something is, in fact, who or what it is declared to be". Whilst other definitions are possible, this is the one that most relates to EMSA's first layer of protection to the MarApps.

The process of authenticating a given person (henceforth referred to as a "user" of the MarApps) is achieved by presenting a place for the user to present his credentials (providing a "user identity" and a password) and then validating the information provided against a repository of known and allowed credentials. This process is achieved in EMSA by Oracle Access Manager (OAM) validating the credentials against EMSA's LDAP.

Correctly authenticated users are allowed access to the next layers of protection while unauthenticated users are never allowed past this first level or layer.

At EMSA, due to the SSO implementation, the user will only be confronted to give his credentials once per session though he will have to pass through the authentication / authorisation process on each request, albeit transparent to him.

3.3. AUTHORISATION

Once a user passes the first layer of protection, i.e. was authenticated, he is subject to the second layer of protection which will only allow the user to access resources (URL's) associated to LDAP groups to which he belongs. At this point, any attempt to access a resource to which the user has not been granted permission will result in an error page being shown indicating that the user does not have permission to access the resource (see following Figure).

Through the extended use of OAM (namely the ability to restrict access to predefined resources (URLs) based upon membership of different LDAP groups), access rights similar to application roles could be enforced without the need for the actual MarApp to implement anything. This mechanism provides a very flexible way of implementing application roles

because there is no need to change the application whenever specific access rules change. There is however the need to update configurations inside of OAM but this is always much simpler and cheaper time-wise than updating code. This mechanism is extensively used for protecting access to the RuleCheck MarApp.

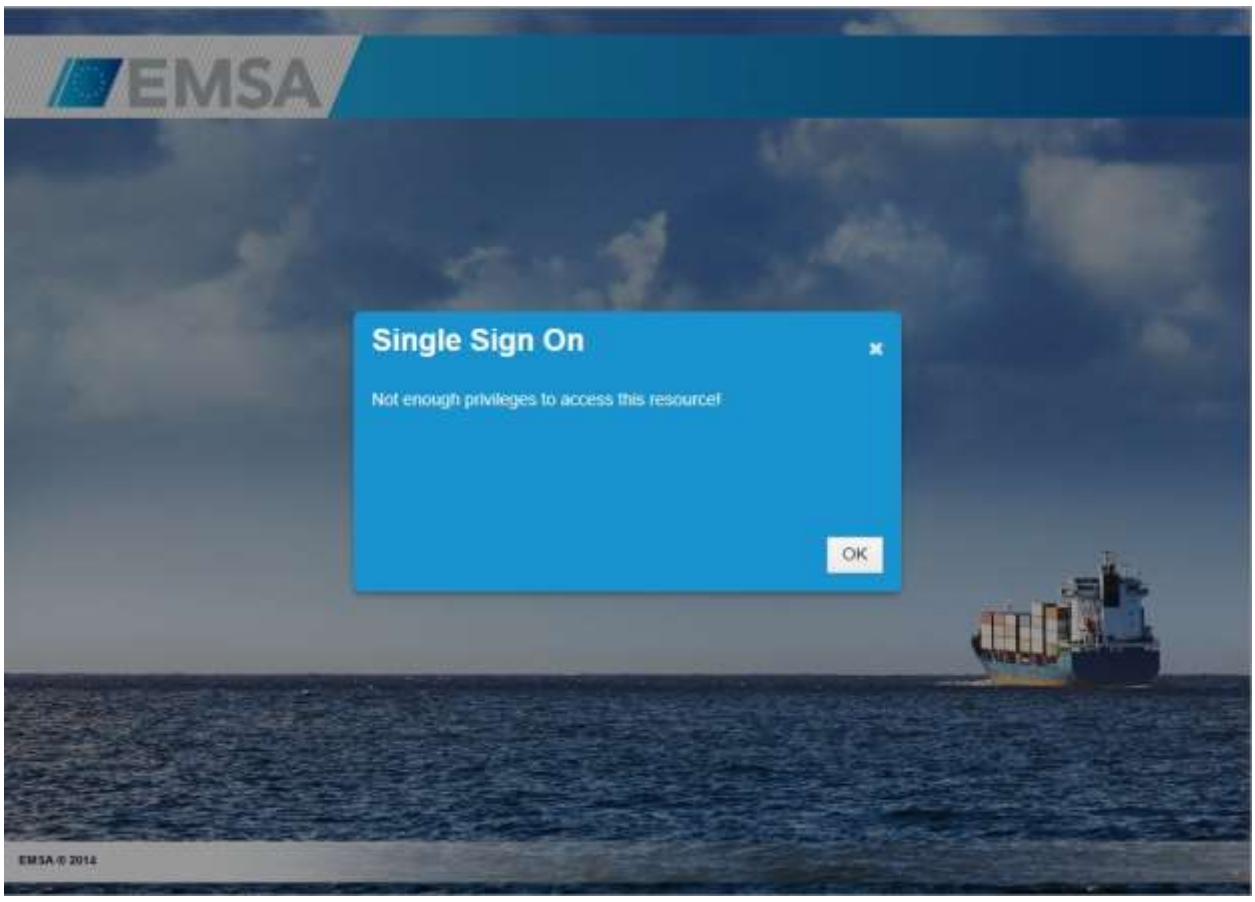


Figure 3: Authorisation denied

Attempts to access resources to which the user has been authorised to do so will result in a transparent intervention from OAM, i.e. nothing specific to OAM will be seen, so the user will not even be aware of existence of the protection layer.

3.4. WEBGATE

Important Note: One very important aspect in EMSA's SSO solution is that only web accesses are considered, i.e. http(s) requests. All other means of access to the EMSA MarApps infrastructure (T3, RMI, etc.) are effectively not protected by this solution.

To enforce the previously mentioned access technology restriction, all protected communication from the MarApps interface (typically a web browser) must go through a proxy/reverse proxy that enforces the first two layers of protection.

In the Oracle technology stack used at EMSA, the proxy/reverse proxy component is called a WebGate (sometimes also referred to as an AccessGate) and is composed of an Apache HTTP Server with, amongst others, Oracle specific modules for communicating/interacting with OAM (obWebgateModule). To obtain a higher degree of service availability various Apache HTTP server instances are running at the same time. We call each instance an SAP (Specific Access Point). Given that EMSA has three environments that are subject to SSO, and various MarApps being accessed through SSO, the total number of configurations

needed makes maintenance a head-ache. To ease this problem the following architecture has been devised.

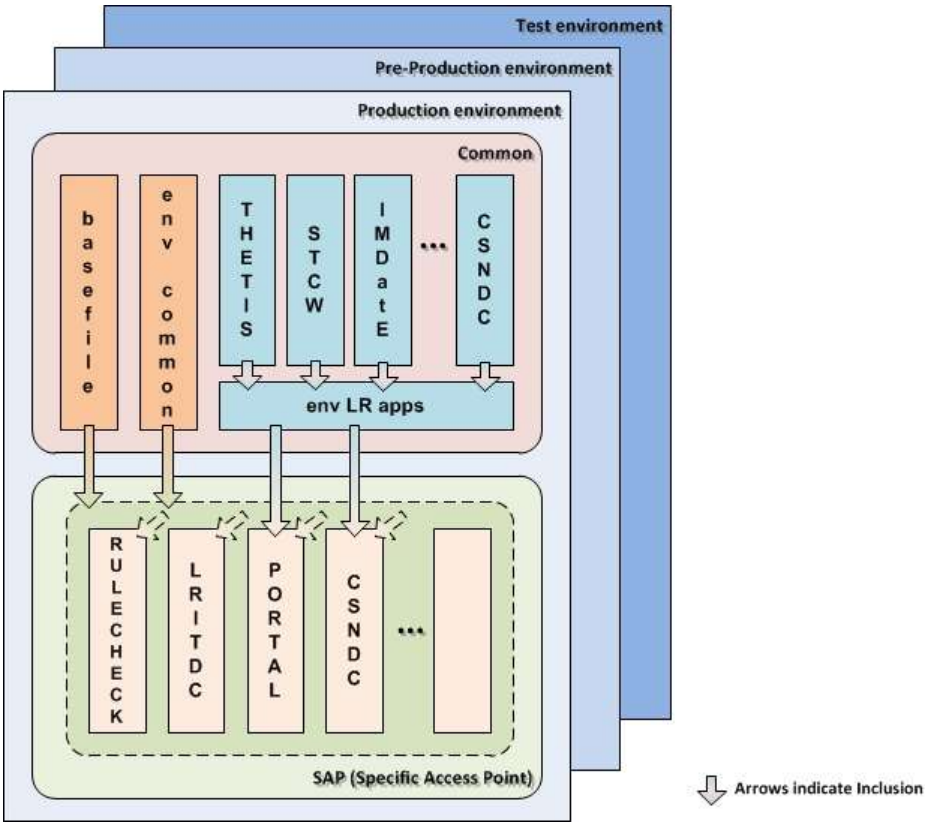


Figure 4: WebGate Configuration Architecture

From observing the previous figure, we can see that in each of the three environments, there are two separate sections: the common configurations section and an SAP (Specific Access Point) configurations section.

The common configuration section is defined only once per environment whilst there are multiple SAPs per environment (not necessarily the same ones in all environments).

3.4.1. SAP Configurations

There have already been a few mentions to an SAP (Specific Access Point) in previous sections of this document but, exactly what is an SAP?

EMSA provides various MarApps to the user community. Some of these are stand-alone apps and some are integrated inside an enterprise portal (Liferay Portal), but all MarApps are web based and thus have a specific URL for being accessed. The unique URL base is what EMSA calls an SAP.

EMSA's production environment contains various SAP, each having its own instance of an Apache HTTP server running. This means that at any given time maintenance can be performed on one SAP while all others are still available/running. Whenever applications share a common access point, i.e. MarApps that are deployed in the Liferay Portal, interventions done to that SAP will obviously affect all those other applications.

The advantages of having SAP are:

- Avoiding unavailability of non-related access points;
- Greatly reducing the amount of work necessary to maintain WebGate configurations by maintaining logical aggregations.

3.4.2. Common Configurations

After having extensively analysed all the configuration files for all MarApps in all environments, a common set of attributes/definitions was identified. To ease the maintenance burden, all the common values were brought together into a single file and explicitly included in each SAP configuration file. Furthermore, each SAP file sets various "*variables*" that are referred to in the common files. This mechanism allows for the maximum re-use of configurations not only across different SAP but also across different environments as well.

Further details can be found in the complete un-abridged version of this document.

3.5. ORACLE ACCESS MANAGER

Earlier in this section mention has been made to authentication of users and authorisation of accessing resources (URLs). The Webgate has been mentioned as being the filtering point for both authentication and authorisation. While this is true, the Webgate is not the system component that implements both functionalities. What it really does is, for each request, question the Oracle Access Manager (OAM) to see if the user is correctly authenticated and if he is authorised to access the resource. If so, the proxy/reverse proxy rules are applied. If not, the user is redirected to a specific page indicating that access rights are denied (if not authorised) or to the login page (if not yet authenticated).

3.5.1. Access Policies

At EMSA, we use the term "Access Policy" to describe the set of configurations needed by OAM to validate access to a specific resource.

Policy Domains

A top-down view of OAM shows the Policy Domains to be the highest level of the configuration structure. Each Policy Domain is a logical aggregator of a set of rules that can be applied to a set of resources (definitions on each of these terms follows). It facilitates management by allowing us to focus on a specific set of logically related rules/resources while permitting the high-level operations of Enabling and Disabling the rules/resources, all at the same time.

Resources

The word resource has come up a lot in this document and it has always been associated with URLs. It is not too farfetched to say that there is an (almost) direct relation between the Resources configured in OAM and the proxy pass rules defined in the Webgate.

Authorisation Rules

An authorisation rule is, as the name implies, a set of rules that define the conditions under which authorisation is granted.

Policies

This is where everything previously mentioned comes together (and is the inspiration for EMSA's nomenclature of "Access Policies"). In a nutshell, this is where the Resources for the policy domain are grouped together with specific authorisation rules.

Examples of policies for a given MarApp can be "Public URLs" and "Private URLs". The resources associated with the Public policy are typically a welcome page in non-portal applications or public portlets in Liferay portal supported applications. Access grants for these types of policies are typically "Allow All".

Associated to a Private policy, we will find resources that are of a more sensitive nature therefore needing protection. With these policies an authorisation scheme is normally used as is an authentication rule.

3.6. RBAC IMPLEMENTATION IN THE EMSA INFRASTRUCTURE

In the scope of the Single Sign On / Identity Management project, we can state that all the applications to be considered are Web Applications. Furthermore, we can also state that all these web applications are to be run under a common "umbrella" which is a Portal environment which will run on Weblogic JEE (Java Enterprise Edition) Application Servers. The Portal environment used at EMSA is based upon the Liferay Enterprise Portal implementation. An LDAP Server supports both the Portal as well as the web applications.

We will now describe how each piece of infrastructure implements/uses the previously mentioned RBAC concepts (basic definitions and relations).

3.6.1. LDAP

An LDAP server allows for the creation of a tree structure of Distinguished Names; DN's in LDAP terminology. It does not directly implement the notion of User Groups or Roles (or even Users for that matter). However, using the DN syntax, one can just about map anything inside the LDAP tree structure. Roles and User Groups can be obtained by associating specific attributes to a DN (whose direct meanings can be interpreted as a Role or User Group) or they can be obtained by answering questions like "in what groups X is a member of" for Roles or "who are the members of that group" for User Groups.

The semantics of use of LDAP at EMSA are:

- The "top level" of the structure having beneath it:
  - The **groups** concept, under which will exist the representation of specific applications (or parts of and extensions to applications).
    - Inside (or underneath) a specific application group should come the actual names of meaningful groups.
  - The **users** concept, under which the **users** branch, two organizational units are possible:
    - Inside the **people** branch are all the physical application users
    - Inside the **system** branch are the system administrators or external systems

- Since the concept of a role is not directly implemented in the EMSA semantics, such a concept should be achieved by associating users to groups through the **member** attribute. By using the first question previously described ("in what groups X is a member of"), one can conclude that in this way it is possible to infer **roles** from this structure (assuming the name of the role is the same as the name of the group for ease of use). The only "restriction" applied here is that the name of the role be the same as the name of the LDAP group supporting the role.
- Applications that require only global authentication should create a group named **members** under the applications own group name and then associate the actual users with this group.
- Applications that need to implement role authorizations should associate the users with the name of the group that represents the desired role.

3.6.2. Liferay Enterprise Portal

The Liferay portal implements the following concepts: Communities, User Groups, Roles and Users. Likewise, the portal implements the concept of a page which we will consider as a

resource in our RBAC model (or Functionality if you like). We will now have a look at each individual concept and discuss it in more detail.

- Users – In Liferay, a User represents a person and has a set of attributes. While it is possible to directly associate Permissions to Users, it is highly recommended not to do so as there are other ways to allow access to resources. There is a "one-to-one" relation between the users in Liferay and the users created in LDAP (even though it is possible for users to exist on only one of either side of the relation).
- User Groups – As the name suggests, this is an aggregator for joining Users. It allows a means for performing some operations on a variable number of users without having to do the same actions on each user individually. Whilst it is possible to assign Permissions to User Groups, as it was for users, this should also not be done. Like the relation between Liferay Users and LDAP users, there is also a "one-to-one" relation between Liferay User Groups and LDAP groups.
- Roles – A role is a way through which Liferay will grant user access to certain resources. A role is logically connected to a User Group (by associating the User Group to the Role) and should maintain a similar name to facilitate human reading/interpretation. This means that any User belonging to the User Group associated with the Role will have access to the resource protected by the Role. In this case, there is no direct connection between a Liferay Role and LDAP even though a logical association may be made through the similarity in the names.
- Sites – In Liferay, a Site is created to allow various Pages (we have called them resources in previous bullets and they are the Functionalities in the RBAC model) to be joined together thus providing a single point of configuration for a specific interest. Whenever access restrictions need to be applied (such as in the private pages of a site), Roles can be associated to a Functionality (Page) in a Site.

We have defined some basic concepts on the RBAC model. We have also explained how this model fits into the EMSA infrastructure. The next section will be about defining the requirements for provisioning users in the EMSA infrastructure for the Maritime applications.

3.7. DEPLOYING APPLICATIONS WITH SINGLE SIGN-ON

Integration with AccMng and SSO can be a simple or a complex task, depending on the type of applications to be integrated.

Simple applications may be integrated without any changes. In this case, AccMng only grants or denies access based on the application URL.
For more complex applications or applications with more demanding access rules:

- Some applications will never need to directly interact with AccMng (such as Thetis, STCW, etc.) because they are executed under the EMSA Portal and/or will obtain the necessary information by using JAAS. The integration sequence diagram in chapter 3.7.1 details the EMSA's Portal integration with AccMng.
- It is likely that other applications may need to be modified to be integrated with AccMng. Chapter 3.7.2 documents how these changes can be done by using a "generic" application such as the Java Pet Store reference application as a "Guiney pig".

3.7.1. Portal integration

The next figure shows a sequence diagram representing the EMSA's Portal integration with AccMng:

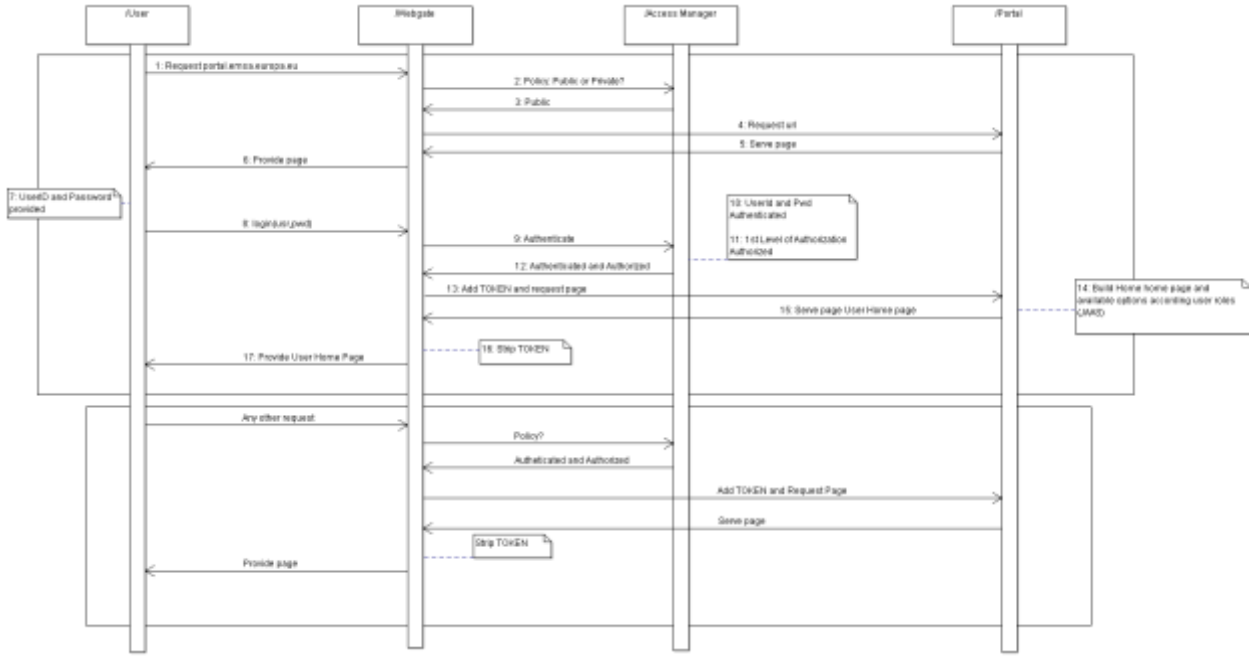


Figure 5: Integration Sequence Diagram

3.7.2. jPetStore

In the EMSA test environment, a well-known reference application – the Java Pet Store – has been deployed that allows for investigation and development of the Single Sign-On solution. One of the goals of deploying such an application in this environment was to assess the difficulties involved in adapting a web application to the Single Sign-On system.

Before going into the details of the necessary changes, we will first explain how the "normal" (unchanged) application works. The Java Pet Store application simulates an on-line shop for selling animals. There is "public" access to the application in which you can browse the existing information and you can even put items into a "shopping cart". If you decide to checkout your order, containing items in the shopping cart, you will have to log-in to the application to be identified. Only users that have been previously registered (provisioned) to the application may checkout orders. Likewise, if you wish to change your user attributes (password, address, phone, etc.) you must also be logged in.

Pre-emptive Authentication

A first interesting approach, while still not the desired one because of not fulfilling the previous "public user" functional requirements, will allow us to demonstrate how to perform authentication through Single Sign-On with minimum changes to the application. In this first approach, the whole application has been registered as "protected" in OAM (Oracle Access Management). This has the effect of the user/password being requested even before the first screen of the application is shown. After the initial logging in to OAM, there is no further need for identifying the user. If a user had already been authenticated in OAM prior to accessing any application screen, he will not be prompted to do so again (Single Sign-On). Please note that the only noticeable change in the application is the fact that the login form is never shown to the user.

Technical Considerations

We have indicated that the jPetStore application is now performing Single Sign-On with minimal changes to the application. We will now proceed to explain the actual changes made.

Three URLs were intercepted (the signonForm, the checkout and the editAccountForm). All three of these URLs have now been internally (internal to the server) redirected to the sign-on URL with additional parameters for the username and password. There are two comments to be made about this URL: first – it is always just internal to the server so there is no problem in sending the username and password as http GET parameters because the internal redirection can never be intercepted, and second – due to the fact that the user's password is never known outside of OAM, we need either to pass the username twice (serving as password) or pass a constant dummy password. This must be consistent with the provisioning process followed.

Public and Private access to the application

As we have previously stated, the pre-emptive authentication scheme is not our target. As such, we now need to make some changes to the OAM to be able to comply completely with the full functional requirements. It is important to point out that there will not be the need to make any more changes to the jPetStore application, but the previously performed changes are still necessary for this stage.

Technical considerations for granting public access

Because of the previous section, the jPetStore application is a protected resource which will require user authentication to be accessed. However, the functional requirements state that there is a part of the application that has public access.

In Oracle Access Manager, access the Policy Manager Application. Under the "Private URLs" policy domain, we will add another policy to the ones already existing in this domain. We have called this new Policy "JPetStore Public" and it consists of an http policy on GETs and POSTs, for all resources and all host identifiers, with the "/jpetstore/…/*" URL pattern.

For this policy to work correctly, the "Authentication Rule" associated to it must be that of "Anonymous Authentication" without any specific "Actions".

Once these changes are made no more user authentication is needed to access the application. There should now be no Authentication Form presented to the user whenever he accesses the jPetStore application, whatever the operation performed within. This, however, is not what is intended as the user will now have to perform an application login (answering to an application login form – not the OAM one) whenever he tries to access the "private" area of the application (accessing the user account or checking out an order).

Final notes on configuration

We have had also the need to configure another policy, the same as the previously mentioned "JPetStore Public", associated to the public URL for the application "/jpetstore".

One other important aspect to consider is the need to change the default session identifier. If the application is implemented using Java technology, change the default session identifier from JSESSIONID to something different (unique to the application), i.e. JSESSIONID_petStore. If you do not make this change, there is a high probability that there will be "session corruption" if more than one Java application is protected by the same Access Manager.

3.8. LOGGING OUT OF SINGLE SIGN-ON

A first-hand premise of SSO is that once a user is authenticated (in any given session), he will be able to access any EMSA Maritime Application to which he is authorised to do so, this without having to re-authenticate himself. The EMSA MarApps must be prepared for the integration with OAM to allow automatically signing in a user and thus achieving an SSO solution.

One often overlooked aspect of an SSO system is that of logging out. Under the assumption that a valid session is in place, a user accessing an application that he has access rights to, will be automatically able to see the respective application (without having to present his credentials again – remember there is a valid session). When a user decides that he does not want to continue accessing a given application, he would normally "logout" from that given application and continues to use any other application that he so wishes. However, due to the automatic nature of SSO, whenever the user re-accesses the original application from which he previously logged out, he will be automatically logged in (due to the auto-login capability of SSO) and will be given the perception that he effectively never logged out. In practical terms this means that the operation of logging out is superfluous unless it is applied to ALL applications that the user was accessing under the current session.

If a global logout solution is not applied, a user can, at any time, simply close a browser tab without logging out of an application as the result is the same as logging out and being automatically logged in again. Please note however that if closing a browser tab results in the end of a browser session (if the tab is the only one open on the browser and no other browser windows are open, for example), then the user will have to log in again if not for any other reason that the browser will not use the same session again when it's re-opened. This is a situation which the user should avoid as the session may still be active in the applications and be subject to session hijacking.

EMSA has chosen to implement a "Single Sign-Out" precisely for the previously mentioned reason of logouts, on their own, being superfluous.

3.8.1. Technical implementation of a global Logout

The implementation done at EMSA is that of once a logout URL is selected (from any of the SSO integrated applications), OAM will intercept the call and start a process of invoking the logout URLs of all the applications to which the user has accessed (been logged in to). After all the application logout URLs have been invoked, OAM will proceed to terminate its own session, thus effectively logging the user out in a safe way.

Each Maritime application that has been integrated with SSO should be prepared to logout correctly upon request.

One final consideration associated to each application needs to be assessed and that is the existence of a logout URL for the application.

3.9. PASSWORD MANAGEMENT

Besides granting access to resources, a Single Sign-On solution has one other major task, that of managing user's credentials or passwords. The managed credentials obey to certain conditions set out by a password policy. We'll explain how credentials are managed at EMSA and the password politics adopted at EMSA in the following sections.

3.9.1. Change Password / Lost Password Management

The EMSA IdM platform is currently responsible for the Password Management actions encompassing several different functionalities. This document only refers to two specific functionalities, change password and lost password.

The SSO solution for managing passwords adopted at EMSA started with an out-of-the-box solution proposed by Oracle but was deemed as inadequate and a new bespoke solution was developed by Oracle.

Change Password

The original implementation allowed *userId* enumeration because the "Change Password" required the user to insert a valid *userId* before going to the actual page to change the password. The navigation was done using a link that was available in the Login screen before the user was authenticated.

Placing this link in a private area has solved the problem. As private areas are only accessible after user authentication, it assures that the user meets the conditions to change his password.

Therefore:

1. The "Change Password" link was removed from the original Login screen;
2. A Link to the "Reset Password" functionality is now available in the "My Information" page provided by IdM to all Maritime Applications. Using this common IdM page avoids the need of changing the Maritime Applications that aren't deployed under EMSA Portal.

Reset Password

IdM V2 now supports a concept of resetting a user's password. This functionality can be accessed when editing an account by selecting the "Reset Password" link. Correct authorization is executed in the implementing code to verify if indeed a user can or not change the password for the account requested (for example, at this time a National Service Administrator – or lower – cannot change passwords for accounts). The basis for this functionality is the "Lost Password" implementation (described next) with the difference that the password introduced can only be used once (to effectively enter the system and change the password to something only known by the end-user) and also the auditing information registered is very clear that the action was done by an administrator and not upon request of the end-user.
This functionality is mainly useful when the "Lost Password" cannot be executed because an invalid email is defined for an account, or when the email is generic for multiple accounts.

Lost Password

A 2-step procedure based on a One-Time generated URL replaced the original Challenge Questions mechanism for the "Lost Password" functionality.

The "Lost Password" function is also able to unlock an account (if previously locked) and provides a detailed logging mechanism to allow an easy diagnosis of faulty or doubtful situations and/or audits.

However, it should be noted that currently e-mails are not unique. Usage of shared e-mails might be problematic from the *End User* point of view. Maritime Applications are strongly encouraged to take measures to address this constraint.

3.10. MAP INTEGRATION

A considerable effort has been made to integrate all EMSA Maritime Applications under the same entry point – known as MAP (Maritime Application Portal).

3.10.1. MAP login Process

MAP has the login screen being directly integrated inside the Portal. With MAP, while the user is still not authenticated he will see a login form on the first page of the Portal where he can directly insert his credentials and proceed to authenticate. All subsequent messages (invalid credentials, etc.) will be displayed in the same space giving the user the impression that he never leaves the screen. For compatibility purposes, for those applications still not integrated into MAP (LRITDC for example), a login screen similar to the MAP layout will be displayed. This will be discussed in the next sub-section.

Figure 6: MAP integrated Login

3.10.2. MAP Access Policies

To cope with this integration, in OAM an URL resource was created "**/mapLogin**" that is associated to a Policy named "**MAP Login**". This policy continues to have *Form Based Authentication*, redirecting to the following URL in case of failure:

- /web/guest/home?p_p_id=login_WAR_emsamaploginportlet&p_p_lifecycle=0&_login_WAR_emsamaploginportlet_failed_login=true

3.11. JSON LOGIN

Originally all EMSA MarApps were web applications that were accessed via internet browser. Over time, due to business and technological advances, some of the MarApps are (also) accessed via dedicated applications running in mobile devices. Two such examples are *Thetis Mobile Application* and *IMS Mobile*. Even though the underlying technology is different in both cases, what we describe next is valid not only for these two cases but also for any other application that wishes to use the same strategy.

3.11.1. User Authentication

In at least one of the cases described previously, i.e. *IMS Mobile*, despite being created as a stand-alone application; it still has the need to access business services supplied by EMSA's infrastructure. Basically, this means that a person using the application will have to identify himself as a recognized user, both to allow actual access to the application as well as to provide boundaries for what information the user can access. As such, the user must be authenticated against EMSA's infrastructure and later authorized to access certain functionality or view determined data. The easiest way to have a clear perception of the user and identify his access rights is using EMSA's SSO infrastructure.
To achieve this goal, EMSA provides a very simple "pseudo web service" that accepts as input the user's identification and his credentials. The information is posted to a URL that processes the information and effectively logs the user in returning success or not logging the user in and returning error. The return information is in the JSON format.

Login URL

The URI to access pseudo web service and attempt a user login is /mobileLogin. Please note that this URI only accepts POST requests and should contain two variables: *userid* that effectively contains the user's id and *password* that will contain the password for the user's account.

Return values

Under normal circumstances, once a POST has been executed to the aforementioned login URI, one of two things can happen:

- The user is authenticated correctly in which case a JSON message consisting of `{ Status: "success" }` is returned;
- The user is not authenticated correctly, or the actual userId does not exist, in which case a JSON message consisting of `{ Status: "error" }` is returned;

The reason this service has been labelled as a "pseudo web service" is because there are certain conditions that will trigger an HTML response instead of a JSON response thus defying compliance to the definition of a web service. The causes of return of such HTML pages are enumerated below. Please note that all these responses should be treated as the user not being logged in. The possible causes are:

- The user is locked out due to having failed his password too many times;
- The user's password is about to expire so a warning of such is sent from OAM;
- The user's password has expired, and a new password should be set;

These cases should be dealt with/resolved by normal access to the SSO login page via a browser.

3.11.2. User Authorization

Once the user is correctly authenticated, the MarApp using the JSON services can obtain information on the user by invocation of the corresponding services described in 4.4.2 User Information Web Service (or "PULL" Model).

4. Identity Management

4.1. EMSA BUSINESS VIEW ON IDENTITY MANAGEMENT

The first version of Identity Management implemented at EMSA was mainly based on an RBAC model (Role Based Access Control) with the user's attributes being spread out in vertical silos (i.e. the MarApps). As more and more MarApps became integrated with IdM, it became evident that there was a set of common attributes that should be the same (instead of the existing value per MarApp model). It was also apparent that an alternative structure to the RBAC model would be beneficial for some MarApps. When it became an absolute necessity to upgrade the underlying platform to a newer version, the opportunity was seized to execute these changes. The latest version of EMSA's IdM now has a common set of attributes per user account as well as providing support for a set of Business entities as described in the following sub-chapters.

4.1.1. Service

A **Service** is a logical entity that represents a set of (one or more) Business Functions typically implemented by an application[3] (MarApp). In the context of the account management, it facilitates the logical discovery of a list of Profiles by filtering those visible or available for choosing. One example is the Thetis Service which has all the Thetis Roles mapped as Profiles and subsequently associated to the Thetis Service.

4.1.2. Profile

A **Profile** is a group of one or more Roles logically combined or aggregated together such that they can be assigned/de-assigned to a User Account, all at the same time. It should be considered as a very high-level logical abstraction of a job function executed by a user.

4.1.3. Role

In the context of EMSA's IdM, a **Role** is a low-level entity that is interpreted in one of two ways, depending on the MarApp or system supporting the role.
For some MarApps (such as Thetis or STCW) a Role is a logical definition of the function assumed or part played by a person or thing in a particular situation. An example of this is a THETIS_INSPECTOR that is a person that has the function of performing inspections of ships according to the PSC regulations. One other example is an LRITDC_ADMIN that is a person that manages the LRITDC MarApp.
Another possible interpretation for the Role is to consider it as a group of permissions that grant or deny access to specific resources. Roles facilitate the assignment of multiple permissions to a User Account. Please note that Permissions themselves are out of scope of IdM. An example of this interpretation is the role VIEW_ABM whose description is "View ABM Alerts". The intent behind this role is to allow a person to view ABM alerts and not that of having a function of spending the time viewing ABM alerts.

[3] Please note that a Service can be implemented by more than one MarApp but in those cases there is always a principal MarApp providing the base functionalities. Please also note that a Service can be implemented via a horizontal platform or system such as Liferay Portal or LDAP

4.1.4. Country/Institution

In the context of EMSA's IdM, the **Country/Institution** defines the "Nationality" (in a broad sense) of a User thus allowing the establishment of an area of control for a *National Administrator* (see 4.2 Security Model). Please note that in EMSA's context, an Institution - such as EFCA for example, is also considered at the same level as a Country.

4.1.5. Organization

At EMSA, the concept of an **Organization** is a sub-entity of a Country or Institution. The Organization a user belongs to is used to establish not only an area of control of a *Local Administrator* (see 4.2 Security Model), but also to filter Profiles and Operations available to be assigned to accounts.

4.1.6. Operation

In EMSA's IdM it's possible to assign **Operations** to an account. In Business terms, an Operation defines an Action that is available to a User in a given context (MarApp). Not all MarApps support Operations, and IdM is completely agnostic to their values and meaning. The list of Operations available to a given user is dependent on that user's Organization.

4.2. SECURITY MODEL

EMSA's IdM is the repository for the account information for users, as well as accumulating as a repository for generic access information to be used by MarApps. It also provides services to access these sets of information. As such, IdM is itself an Application and needs to have its own set of business rules to regulate who can do what in the IdM application. In IdM, the foundation for this regulation is the Security Model which establishes the management relationships (who is entitled to create/edit other users) and the permission rules (which serve as filters for limiting who a user can administer) or, said in another way: The Security Model defines who can do what in a hierarchical way.

The EMSA Security Model has 5 hierarchical levels. From the most privileged level to the least, these are:

1. **EMSA Administrator**
   Identity Manager *super users*. Users belonging to this level are entitled to manage ***all user accounts without restrictions*** and they also have privileges to access some normally restricted IdM functionalities. "EMSA Administrator" level can only be assigned to a person belonging to EMSA and is normally limited to a very small number of people as it implies knowledge of a specific skill-set.

2. **EMSA Service Administrator**
   Identity Managers for a specific Service. Users belonging to this level are entitled to manage ***user accounts related with a specific Service*** (i.e. the services defined as those he is administrating). "EMSA Service Administrator" can only be assigned to a person belonging to EMSA and should be limited to a small number. It should be noted that a single person can be associated (i.e. have this level) with more than one service.

3. **National Service Administrator**
   Identity Managers for a specific Country/Institution relating to a specific service. Users belonging to this level are entitled to manage ***user accounts that are simultaneously related with the Administrator´s Service and the Administrator's Country/Institution***. "National Administrator" level can be

assigned to any user of a specific Country/Institution even though at the business level there is normally a very limited set of people that possess this privilege.

4. **Local Service Administrator**
   Identity Managers for a specific Organization inside a specific Service and Country/Institution. Users belonging to this level are entitled to manage ***user accounts that are simultaneously related with the Administrator´s Service, Country/Institution and Organization***. "Local Administrator" level can be assigned to any user of a specific Country/Institution for a given Organization.

5. **End-user**
   End-Users have the most limited set of Identity Management privileges. They are only entitled to modify a limited set of their own personal attributes (i.e. the ones which are common to all applications).

It should be noted that not all Maritime Applications contemplate the use of all levels. Most notably the **Local Administrator** is rarely used by most applications.

4.2.1. Security Model Level Correspondence to Application Roles

One common misconception that occurs relating to IdM is the assumption of an implicit relationship between the EMSA Security Model and the Maritime Application functional roles. **This implicit relationship does not exist**. Any given user can be, for example, an end-user within an application and simultaneously be an Administrator (EMSA or National level) within IdM (for that same application). There is no mechanism imposing any limitation whatsoever. However, it is common for applications to request the establishment of a relationship of their internal application roles to certain security model levels **explicitly**.

The explicit relationship establishment is done through role mappings, i.e. each role is assigned a security level value. This means that whenever a given application role is assigned to a user, he will "inherit" (be automatically assigned) a certain security model level. One example of such a mapping is the "Thetis System Administrator" role has a security level value of "EMSA Service Administrator". This means that whenever a user is assigned that Role, he will become an "EMSA Service Administrator" for Service Thetis (as this Role is associated to this Service).

In the end, it is important to retain that management inside IdM is completely independent of any form of management within any given Maritime Application.

4.2.2. Accumulation of Levels

An important misconception is that an account has one (and only one) security level. While it's true that if a user has various Roles assigned to him (via Profiles), he will have the highest security level of all those Roles, this must be seen in the context of the Service to which the roles belong. It is possible for a user to be, for example, a National Service Administrator for one given Service while still being an End-User for another distinct Service.

4.3. IDENTITY MANAGEMENT FUNCTIONALITIES

For a User to have access to a MarApp with the appropriate permissions, IdM must handle User Account Management, including Provisioning of User Attributes.
While some operations of the application are performed following an automated process, most require either the intervention of or the initiation by a user. The following chapters demonstrate the various aspects of Identity Management.

4.3.1. Reconciliation

The Reconciliation functionality of IdM is responsible for importing data from external systems, necessary for configuring a User Account. Examples of reconciliation of data is the list of Countries/Institutions from CBR (Country Base Registry), Organizations from COD (Central Organization Database), and low-level information used for provisioning from the Staging Area Database.
It should be noted that IdM is the authoritative source for User information.

4.3.2. Account Management

Account Management is the name given to the set of actions that may be performed on a User Account to Create, Remove, Update/modify, Delete/disable. Besides the typical CRUD functionalities available, as part of IdM it is also possible to Search for Accounts, view the relationships between Services / Profiles and Roles, view the auditing information on changes made to accounts as well as recovery of failed provisioning attempts.

4.3.3. Provisioning

The act of provisioning is the process by which IdM provides the updated User Account data to all the MarApps/Systems the affected User has access to (possesses a Profile/Role for). Please note that this effectively corresponds to the "PUSH" model described in 4.4.1 Provisioning Applications or "PUSH" Model.

4.3.4. Other Administrative Functions

In this category are other operations not included in the other groups and are available only to the highest Administrator level, such as Reporting, Exports and Bulk Load.

4.4. IDENTITY MANAGEMENT INTEGRATIONS

EMSA's Identity Management system (IdM) is fully integrated within EMSA's applicational infrastructure. This effectively means that it can be accessed via Single Sign-On like any other MarApp/System. It can send information (i.e. invoke services or return data via invoked services) to other existing MarApps/Systems and finally, it can also receive requests to moderate its behaviour (i.e. provide functionalities upon request). The following chapters describe some of these aspects.

4.4.1. Provisioning Applications or "PUSH" Model

One of the goals of having an Identity Management solution in EMSA is to have a common way of provisioning users to applications. This essentially means that whatever can be found common to all possible applications should be stored locally in IdM and thereafter provisioned to each MarApp that the user is effectively a member of (i.e. having a relevant Role in that MarApp).

EMSA's Maritime Applications are provisioned by IdM to contain user information. This mechanism can be roughly described as being a "PUSH" mechanism in that EMSA's Identity Management system effectively sends information on new users (or changes made to existing users) to the appropriate systems/MarApps for which this information is relevant. The way this is done is through invocation of a series of established Web Services available

in the MarApps (or eventually using another form of API such as an LDAP connection). IdM attempts to guarantee that every system/MarApp is kept up-to-date with the latest information on a user, be it personal attributes such as first name, etc. or authorization information such as Profiles (via Roles).

4.4.2. User Information Web Service (or "PULL" Model)

It should be noted that in the EMSA eco-system of Maritime Applications and horizontal platforms, the "PUSH" mechanism may not always be the best solution for user management. There are cases in which the actual MarApp is significantly (or even completely) agnostic about users. One such example is STCW which has the need for "knowing" users and their personal attributes (such as their "Country" for example) but does not actually keep any information about them. Another much more radical example is RuleCheck that, in its current form, has absolutely no concept or knowledge of users. RuleCheck's content is served to users in a differentiated model by strategic use of the Access Management component of IdM (the OAM) allowing or denying users to see certain content. Still another example is the SEG (SafeSeaNet Eco-system GUI) that only needs to know what the accessing user can or cannot do and see (i.e. his Profiles/Roles and Operations). This necessity led to the establishment of a new architectural model, the "PULL" model. The "PULL" model is none other than a service supplied by IdM allowing any MarApp to request information about a given user.

UserInfo REST Web Service

Currently the "PULL" model is implemented through a REST Web Service that is invoked via a normal HTTP GET call passing the user's identification and returning information in the JSON format.

The UserInfo REST Web Service can only be called from within EMSA's infrastructure so no data leakage can occur to entities outside of EMSA. The context path of the service is not available through any URL that can have public access. There is currently no assumption made about the user invoking the service, so no authentication is done.

The UserInfo Web Service will typically return one of two possible sets: a null message if the user ID passed as the last parameter does not exist in IdM, or a JSON message containing information on the user ID passed.

4.4.3. Accessing IdM Functionalities via direct URL

EMSA needs to access OIM directly from links placed in some applications, namely MarApps and Liferay Portal. A bespoke module has been developed to allow such direct access.

Search User

The URL for "jumping" into IdM directly in the Search Users Functionality is

/identity/faces/home?tf=**SEARCH_USERS**

Through this URL authorized users can execute a search for one or more accounts and then proceed to execute other actions (view, edit, etc.)

Create User

The URL to access the Create User Form is

/identity/faces/home?tf=**CREATE_USER**

Edit User

The URL to access the Edit User Form window is

/identity/faces/home?tf=**MODIFY_USER**&userLogin=**<User Login>**

In the previous link, <*User Login*> must be changed to the correct user identification per call to the edit method.

Edit my account (only fields common to all applications)

The URL to access the Edit my account User Form window is

/identity/faces/home?tf=**MODIFY_USER**

You might notice that this link is similar to the Edit User link except for the fact that no UserLogin identification is provided and as such the account of the user accessing is displayed.

4.5. STAGING DATABASE MANAGEMENT CONSOLE

The purpose of the Staging Database Management Console (SDMC) is to manage the Staging Database Entities along with their relationships, providing also Transaction consistency functionality in order to avoid synchronization incompatibilities.

The following are basic business rules that reflect the architecture design of the Staging Database:

- A Country may be related with none, one or several Organizations.
- An Organization is a hierarchical structure that may have none or one parent.
- An Organization belongs to one unique Country.
- A Profile may include one or several Roles.
- Roles may be used by none, one or several Profiles.
- Each Role belongs to one unique Service.
- An Organization can be related with none, one, or several Operations.
- An Operation can be related with none, one or several Organizations.
- A Provisioning Endpoint may be used by none, one or several Roles.
- A Role may send information to none, one or several Provisioning Endpoints.

The SDMG provides a User Interface for the management of the following entities and their relationships:

- Profile
- Role
- Service
- Operation
- Provisioning Endpoint
- Security Level

- Country
- Organization

In this scope, it is considered that the management of each Entity type includes:

#	Business Requirement Title	Business Requirement Description
1	Search Entities	Search Entities based on attribute criteria. a. Search of each attribute will be based on the Contains clause. b. Search with the AND operator for multiple attributes criteria.
2	Grouping and Ordering	Grouping and Ordering capabilities of the search results.
3	Access functionality from Search	Provide access to Create, Modify, Disable/Enable functionalities from the search results.
4	Create Entity	Create a new Entity and their direct relationships. a. Ability to accept all Entity attributes and create a new entry in the Entity table. b. Unique Code attribute should be composed using a pre-defined prefix. c. Last Changed attribute should have the timestamp of the creation moment of the Entity.
5	Update Entity	Update an existing Entity and their direct relationships. a. Ability to accept modifications to the Entity attributes. b. The Unique Code attribute cannot be modified. c. Last Changed attribute should have the timestamp of the last change of the Entity.
6	Disable/Enable Entity	Disable/Enable an existing Entity and their direct relationships. a. Ability to set and reset the Entity status flag to enabled or disabled. b. Enabling or Disabling a top or hierarchically higher Entity should be reflected in all its direct relationships.
7	Relationship flexibility	Ability to clearly see and directly jump to the management functionalities of all other direct relationships of an Entity.
8	Authentication process	EMSA Access Management should be responsible for the authentication process.
9	Authorization process	SDMC should be responsible for the authorization process. Authorizing user's accounts to access SDMC functionalities should be based on specific roles granting user specific privileges for that effect.
10	Staging Database Consistency	The ability to temporarily store or permanently commit the modifications in order to not have inconsistencies.

ANNEX A – EMSA Customisations on OIM – Oracle Identity Management

To comply with EMSA's requirements for Identity Management, the Oracle Identity Management tool was customized. At the Database level, various entities were introduced. These are listed and described below.

Database Table	Description
Service	Maritime Business Services. A Service represents a set of (one or more) Business Functions implemented by an application (MarApp). Examples of Services are Thetis, IMS - Integrated Maritime Services, EOS - Earth Observation Services
Profile	Business Profiles. A role played by a person within the context of Maritime Applications. Examples are Thetis Allocator, Document Management Reader, LRITDC EU DC Administrator
Role	Maritime Application Roles. An entity that can correspond to an RBAC role within a Maritime Application or in exceptional cases correspond to a permission within a Maritime Application. Examples are Locations Manager, CSD Viewer, View ABM Reports.
Operation	Maritime Business Operations. Within the context of a Maritime Application, an Operation defines an Action that is available to a User. Examples of Operations are Frontex, CleanSeaNet, SafeMed
Country	Registered Countries / Institutions. Define the Nationality of a User and the area of control of a National Administrator. They can be Institutions, Companies or Regional Agreements instead of Countries. Examples are EMSA, Portugal, FRONTEX.
Organization	Organizations. Define the actual Country Organization a User belongs to and the area of control of a Local Administrator. Examples are The Antigua Department of Marine Services, Directorate of Shipping Aruba, Port of Zeebrugge.
Provisioning_EndPoint	Provisioning Points. These are physical systems that receive data from the Identity Management system. Examples are Liferay Portal, LDAP, Thetis.
Security_Level	Security Levels. The various classifications assigned to accounts via Roles assigned that define permissions/allow actions within the Identity Management system. Examples are EMSA Administrator, National Service Administrator, End-User.
Operation_Organization	Operations – Organizations associations. This indicates what Operations are assignable to accounts having what Organizations. Data restriction is relative to the account being edited. All Operations should be assignable to Organization EMSA.
Profile_Organization	Profiles – Organizations associations. This indicates what Profiles are available for assigning to other accounts by the account doing the editing (having the Organization). Data restriction is relative to the account doing the editing. All profiles should be allowed for Organization EMSA.
Profile_Role	Profiles – Roles associations. This indicates what Roles are provisioned when the account is assigned the Profile.
Service_Role	Service – Role associations. This establishes the relationship between a Role and the underlying Business Service that implements the functionalities allowed via the Role.
Role_Provisioning	Role – Provisioning points associations. This establishes the actual physical system that is provisioned when the Role is to be provisioned (via assignment of Profile).
IDM_AUDIT_USR_M	Auxiliary table used for provisioning accounts
IDM_ORGANIZATION_CHILD	Auxiliary table used for provisioning accounts
IDM_PROVISIONING_FAILURE	Auxiliary table used for provisioning accounts
IDM_PROVISIONING_FAILURE_MSG	Auxiliary table used for provisioning accounts
IDM_TASK	Auxiliary table used for provisioning accounts
IDM_USR_AUX	Auxiliary table used for provisioning accounts
IDM_USR_M	Auxiliary table used for provisioning accounts
IDM_USR_M_OLD	Auxiliary table used for provisioning accounts

Database View	Description
IDM_USER_V	Database view that exposes, in human readable format, a complete attribute set of the existing accounts.
IDM_USER_OP_V	Database view that exposes, in human readable format, the Operations associated to the accounts.
IDM_USER_SP_V	Database view that exposes, in human readable format, the Profiles associated to the accounts. It also shows the Services associated via indirect relationship Profiles -> Roles -> Services.
IDM_USER_PR_V	Database view that exposes, in human readable format, the Profiles associated to the accounts as well as the Roles associated to those Profiles.

ANNEX B – Statistical Information in Identity Management

EMSA's Identity Management system is a repository for certain information used within its Maritime Applications. That information is stored in the form of entities.

As of 22nd April 2020, the quantities of such entities are described below.

Production Environment

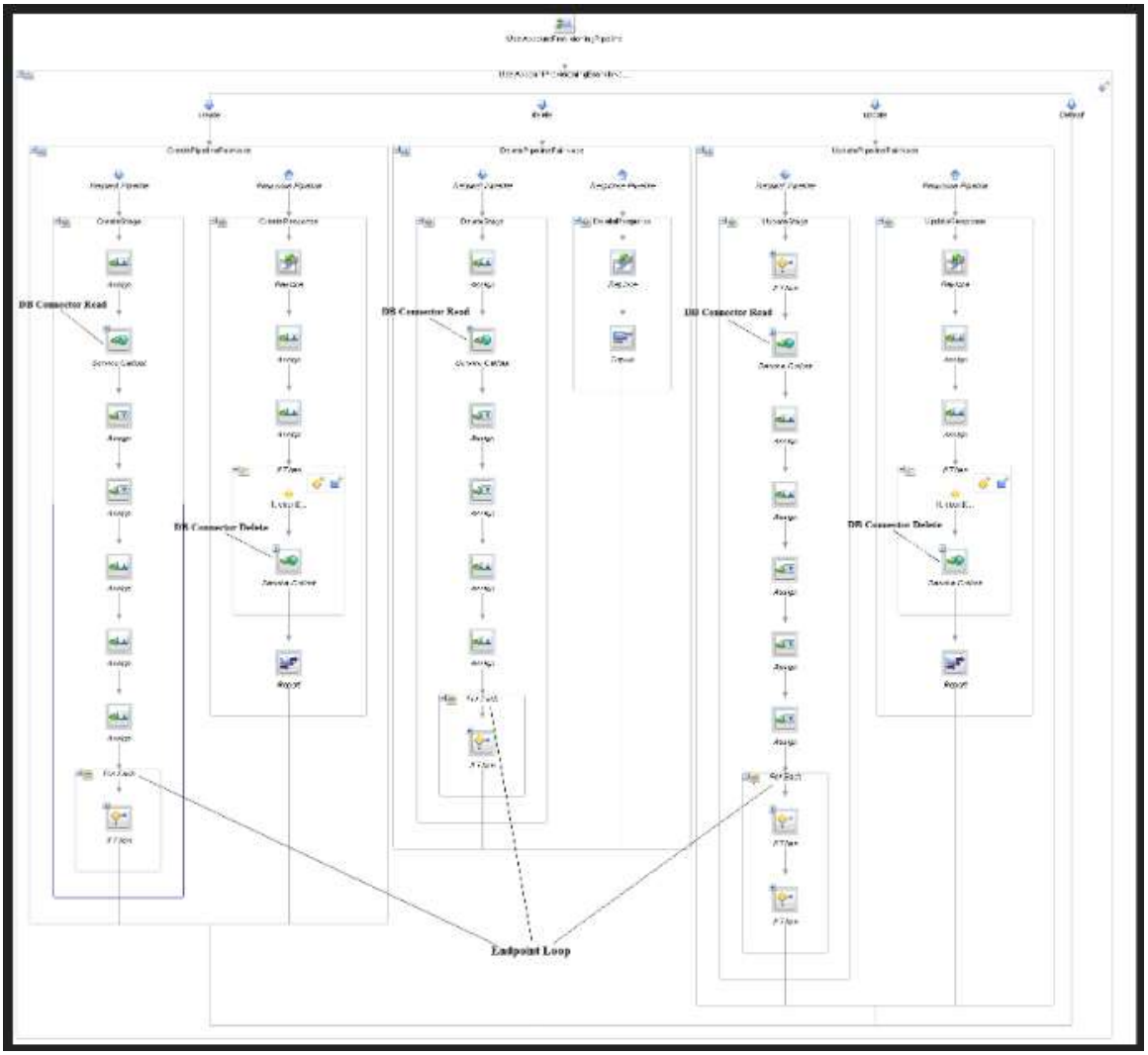
Entity	Quantity
Registered Accounts	13988 accounts
Maritime Business Services	21 Services
Business Profiles	647 Profiles
Maritime Application Roles	687 Roles
Maritime Business Operations	36 Operations
Registered Countries / Institutions	107 Countries / Institutions
Organizations	281 Organizations
Provisioning Points	8 Provisioning Points
Security Levels	5 Security Levels
Operations – Organizations associations	528 associations
Profiles – Organizations associations	2710 associations
Profiles – Roles associations	1383 associations
Service – Role associations	696 associations
Role – Provisioning points associations	1274 associations

ANNEX C – SOA Suite Processes

The architecture chosen for EMSA's Identity Management separates the act of creating/editing an account in the UI tool (OIM + SOA) from the act of provisioning (SOA + OSB) the information to the physical systems (Maritime Applications and support platforms).

When a message reaches the provisioning layer, it does so via the User Account Provisioning Proxy Service. A Connector is responsible for calling the corresponding operation of the User Account Provisioning Proxy Service, which in turn decides for the appropriate flow to be followed. The description of the User Account Provisioning Proxy Service is depicted in the figure below.

Please note that there is an embedded object following the image that can be expanded allowing viewing of the complete process in greater detail.



User Account
Provisioning Proxy

ANNEX D – Web Service Details

This section details what was overviewed in section 4.4.2 User Information Web Service (or "PULL" Model), the UserInfo web services.

The UserInfoInterface Proxy SOAP and Rest services provide the User Account information by calling the IDM-Interface SOA Composite. The SOAP service requires user authorization to perform a successful call and the output is used for reporting purposes. For the Rest service no authorization is required.
The Adapter exposes operations allowing search and extraction of data with the following criteria:

- SOAP
  - User Account Id
  - Status
  - Last Update Date
  - Type
  - Country
  - Organization
  - Service
  - Profile
  - Operation
- Rest
  - User Account Id

The information below is valid for both SOAP and REST services

UserInfoRequest operation of IDM-Interface Composite The search fields for the SOAP service can be combined in a single search with an AND clause only.			
Inputs			
Name	**Mandatory**	**Type**	**Description**
accountId	No	String	The id of the user.
status	No	String	The user status.
lastUpdateStart	No	String	The user's last update date.
lastUpdateEnd	No	String	The user's last update date.
type	No	String	The user's type.
country2Code	No	String	The user's country code.
organizationCode	No	String	The user's organization code.
profileCode	No	String	The user's profile code.
operationCode	No	String	The user's operation code.
serviceCode	No	String	The user's service code.
startRow	No	Number	From which user account should start returning results.
endRow	No	Number	Until which user account should stop returning results.
Outputs			
Name	**Mandatory**	**Type**	**Description**
type	Yes	String	The type of the account.
accountId	Yes	String	The id of the user.
securityLevelCode	Yes	String	The security level code of the account.
status	Yes	String	The status of the account.
disableDate	No	Date	The disable date of the account.
lastUpdate	Yes	Date	The user's last update date.
initial	No	String	The initials of the user.

firstName	Yes	String	The first name of the user.
middleName	No	String	The middle name of the user.
lastName	Yes	String	The last name of the user.
email	Yes	String	The email address of the user.
address	No	String	The postal address of the user.
phone	Yes	String	The telephone of the user.
fax	No	String	The fax number of the user.
alertEmail	No	String	The alert email of the user.
alertPhone	No	String	The alert phone of the user.
categoryType	Yes	String	The authority type of the user.
country	Yes	String	The country of the user.
OrganizationOperationsInfo	Yes	ListOfObject	The organization and operations data.

Name	**Type**
organizationCode	string
organizationDescription	string
operationDescription	string
operationCode	string

SecurityLevel	Yes	Object	The security level data.

Name	**Type**
securityLevelCode	string
securityLevelDescription	string

CountryInstitution	Yes	Object	The country data.

Name	**Type**
categoryType	string
country	string
country2Code	string

ServicesProfilesInfo	No	ListOfObject	The service and profile data.

Name	**Type**
serviceCode	string
serviceDescription	string
profileCode	string
profileDescription	string

RolesInfo	No	ListOfObject	The role data (only REST).

Name	**Type**
roleCode	string
roleDescription	string

Depending on the query being made, the SOAP Service might return a large set of results; to avoid stalling the response, if the number of accounts is higher than an established limit, the SOAP Service automatically splits the result set in different pages.

SOAP Interface:
URL:http://**<SERVER>**:**<PORT>**/IDMExposedInterfacesProject/ProxyServices/UserInfoInterface?WSDL

Where <SERVER> is environment dependent.
Access to the SOAP interface is granted only to Authenticated/Authorized accounts.

REST interface:
URL: http://**<SERVER>**:**<PORT>**/UserInfoInterfaceService/UserInfo/{accountID}

Where <SERVER>:<PORT> is environment dependent.
No Authentication/Authorization is required.

SOAP Example

REQUEST

========

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:typ="http://org.exposed_interfaces.genericws/types">
  <soapenv:Header/>
  <soapenv:Body>
    <typ:userInfoRequest>
      <!--Optional:-->
      <typ:accountId>000-TEST-26</typ:accountId>
      <!--Optional:-->
      <typ:country2Code></typ:country2Code>
      <!--Optional:-->
      <typ:organizationCode></typ:organizationCode>
      <!--Optional:-->
      <typ:serviceCode></typ:serviceCode>
      <!--Optional:-->
      <typ:profileCode></typ:profileCode>
      <!--Optional:-->
      <typ:operationCode></typ:operationCode>
      <!--Optional:-->
      <typ:status></typ:status>
      <!--Optional:-->
      <typ:type></typ:type>
      <!--Optional:-->
      <typ:lastUpdateStart></typ:lastUpdateStart>
      <!--Optional:-->
      <typ:lastUpdateEnd></typ:lastUpdateEnd>
      <!--Optional:-->
      <typ:startRow></typ:startRow>
      <!--Optional:-->
      <typ:endRow></typ:endRow>
    </typ:userInfoRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

RESPONSE

=========

```
<soapenv:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <env:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
    <wsa:MessageID>urn:1d86bd78-c189-11e8-97c6-0050569c0895</wsa:MessageID>
    <wsa:ReplyTo>
      <wsa:Address>http://www.w3.org/2005/08/addressing/anonymous</wsa:Address>
      <wsa:ReferenceParameters>
        <instra:tracking.ecid xmlns:instra="http://xmlns.oracle.com/sca/tracking/1.0">646a055d-9bdb-442e-9737-27b2a7c40e4e-
00069ca5</instra:tracking.ecid>
        <instra:tracking.FlowEventId xmlns:instra="http://xmlns.oracle.com/sca/tracking/1.0">190471</instra:tracking.FlowEventId>
        <instra:tracking.FlowId xmlns:instra="http://xmlns.oracle.com/sca/tracking/1.0">151210</instra:tracking.FlowId>
        <instra:tracking.CorrelationFlowId
xmlns:instra="http://xmlns.oracle.com/sca/tracking/1.0">0000MOLstJm33FGqywZf6G1RFDBL00007T</instra:tracking.CorrelationFlowId>
        <instra:tracking.quiescing.SCAEntityId
xmlns:instra="http://xmlns.oracle.com/sca/tracking/1.0">40004</instra:tracking.quiescing.SCAEntityId>
      </wsa:ReferenceParameters>
    </wsa:ReplyTo>
    <wsa:FaultTo>
      <wsa:Address>http://www.w3.org/2005/08/addressing/anonymous</wsa:Address>
    </wsa:FaultTo>
  </env:Header>
  <env:Body xmlns:wsa="http://www.w3.org/2005/08/addressing">
    <ns2:userInfoResponse xmlns:ns2="http://org.exposed_interfaces.genericws/types">
      <ns2:returnMessage>Rows 1 to 1 from total 1</ns2:returnMessage>
      <ns2:userInfo>
        <ns2:rowNumber>1</ns2:rowNumber>
        <ns2:type>human</ns2:type>
        <ns2:accountId>000-TEST-26</ns2:accountId>
        <ns2:securityLevel>
          <ns2:securityLevelDesc>EMSA Service Administrator</ns2:securityLevelDesc>
          <ns2:securityLevelCode>4</ns2:securityLevelCode>
        </ns2:securityLevel>
        <ns2:status>Active</ns2:status>
        <ns2:disableDate/>
        <ns2:lastUpdate>2018-09-19T13:50:14</ns2:lastUpdate>
        <ns2:personalInfo>
          <ns2:initial>TST</ns2:initial>
```

```
            <ns2:firstName>000-TEST-26</ns2:firstName>
            <ns2:middleName/>
            <ns2:lastName>000-TEST-26</ns2:lastName>
            <ns2:contactDetails>
              <ns2:email>000-TEST-26@emsa.europa.eu</ns2:email>
              <ns2:address>Praca Europa 4 Lisbon Portugal</ns2:address>
              <ns2:phone>123456789</ns2:phone>
              <ns2:fax/>
              <ns2:alertingDetails>
                <ns2:email/>
                <ns2:phone/>
              </ns2:alertingDetails>
            </ns2:contactDetails>
          </ns2:personalInfo>
          <!--The User Country information-->
          <ns2:countryInstitutionInfo>
            <ns2:categoryType>INSTITUTION</ns2:categoryType>
            <ns2:country>EMSA</ns2:country>
            <ns2:country2Code>XX</ns2:country2Code>
          </ns2:countryInstitutionInfo>
          <!--The User organization and operation information-->
          <ns2:organizationOperationsInfo>
            <ns2:organizationDescription>EMSA</ns2:organizationDescription>
            <ns2:organizationCode>ORG_EU00007</ns2:organizationCode>
            <ns2:operations>
              <ns2:operation>
                <ns2:operationDescription>CleanSeaNet</ns2:operationDescription>
                <ns2:operationCode>OPR_CSN</ns2:operationCode>
              </ns2:operation>
              <ns2:operation>
                <ns2:operationDescription>EFCA Atlantic</ns2:operationDescription>
                <ns2:operationCode>OPR_EFCA_ATLANTIC</ns2:operationCode>
              </ns2:operation>
            </ns2:operations>
          </ns2:organizationOperationsInfo>
          <!--The User service and profile information-->
          <ns2:servicesProfilesInfo>
            <ns2:serviceProfileInfo>
              <ns2:serviceDescription>CHD-MARCIS2 - Central Hazmat Database</ns2:serviceDescription>
              <ns2:serviceCode>SRV_CHD_MARCIS2</ns2:serviceCode>
              <ns2:profileDescription>MARCIS2 User</ns2:profileDescription>
              <ns2:profileCode>PRF_MARCIS2_USER</ns2:profileCode>
            </ns2:serviceProfileInfo>
            <ns2:serviceProfileInfo>
              <ns2:serviceDescription>IMS - Integrated Maritime Services</ns2:serviceDescription>
              <ns2:serviceCode>SRV_STAR</ns2:serviceCode>
              <ns2:profileDescription>Access to IMDatE WUP</ns2:profileDescription>
              <ns2:profileCode>PRF_IMDATE_BASIC_VIEWER</ns2:profileCode>
            </ns2:serviceProfileInfo>
          </ns2:servicesProfilesInfo>
        </ns2:userInfo>
      </ns2:userInfoResponse>
    </env:Body>
</soapenv:Envelope>
```

REST Example

REQUEST

========

http://<SERVER>:<PORT>/UserInfoInterfaceService/UserInfo/000-TEST-26

RESPONSE

========


```
{
  "type": "human",
  "accountId": "000-TEST-26",
  "securityLevel":   {
    "securityLevelDesc": "EMSA Service Administrator",
    "securityLevelCode": "4"
  },
  "status": "Active",
  "lastUpdate": "2018-09-19T13:50:14",
  "personalInfo":   {
```

```
    "initial": "TST",
    "firstName": "000-TEST-26",
    "middleName": null,
    "lastName": "000-TEST-26",
    "contactDetails":       {
      "email": "000-TEST-26@emsa.europa.eu",
      "address": "Praca Europa 4 Lisbon Portugal",
      "phone": "123456789",
      "fax": null,
      "alertingDetails":         {
        "email": null,
        "phone": null
      }
    }
  },
  "countryInstitutionInfo":    {
    "categoryType": "INSTITUTION",
    "country": "EMSA",
    "country2Code": "XX"
  },
  "organizationInfo":    {
    "organizationDescription": "EMSA",
    "organizationCode": "ORG_EU00007"
  },
  "operationsInfo":    [
        {
      "operationDescription": "CleanSeaNet",
      "operationCode": "OPR_CSN"
    },
        {
      "operationDescription": "EFCA Atlantic",
      "operationCode": "OPR_EFCA_ATLANTIC"
    },
  ],
  "servicesInfo":    [
        {
      "serviceDescription": "CHD-MARCIS2 - Central Hazmat Database",
      "serviceCode": "SRV_CHD_MARCIS2"
    },
        {
      "serviceDescription": "IMS - Integrated Maritime Services",
      "serviceCode": "SRV_STAR"
    },
  ],
  "profilesInfo":    [
        {
      "profileDescription": "MARCIS2 User",
      "profileCode": "PRF_MARCIS2_USER"
    },
        {
      "profileDescription": "Access to IMDatE WUP",
      "profileCode": "PRF_IMDATE_BASIC_VIEWER"
    },
  ],
  "rolesInfo":    [
        {
      "roleDescription": "ROL_MARCIS2_USER",
      "roleCode": "ROL_MARCIS2_USER"
    },
        {
      "roleDescription": "Access to IMDatE WUP",
      "roleCode": "ROL_IMDATE_BASIC_VIEWER"
    },
  ]
}
```

European Maritime Safety Agency

EMSA secure web application requirements

Version 1.0
Date: 27/09/19

Document History

Version	Date	Changes	Prepared	Approved
1.0	27/09/2019	Publish	EMSA	

Table of Contents

Acronyms	
L1	Level 1
L2	Level 2
L3	Level 3
CSP	Credential Service Provider also called an Identity Provider
OTP	One-time password
SFA	Single factor authenticator
MFA	Multi factor authenticator, which includes two or more single factors
CWE	Common Weakness Enumeration (CWE) is a community-developed list of common software security weaknesses
2FA	Two-factor authentication
ASLR	Address Space Layout Randomization
ASVS	Application Security Verification Standard
DAST	Dynamic
SAST	Static application security testing
OWASP	Open Web Application Security Project
SDLC	Software development lifecycle

0. OWASP Application Security Verification Standard

Secure development is a requirement for any application or component that is integrated into EMSA ICT Landscape. OWASP Application Security Verification Standard is an industry standard maintained by the OWASP foundation that complies with EMSA requirements to verify that specific security measures are in place in the application or code. This document is based on the OWASP ASVS version 4.

OWASP ASVS has two main goals:

- to help organizations develop and maintain secure applications.
- to allow security service vendors, security tools vendors, and consumers to align their requirements and offerings.

0.1 Application Security Verification Levels

The Application Security Verification Standard defines three security verification levels, with each level increasing in depth.

- ASVS Level 1 is for low assurance levels, and is completely penetration testable
- ASVS Level 2 is for applications that contain sensitive data -both commercial or personal-, which requires protection and is the recommended level for most apps
- ASVS Level 3 is for the most critical applications - applications that perform high value, contain sensitive data, contain EU confidential data or any application that requires the highest level of trust.

Each ASVS level contains a list of security requirements. Each of these requirements can also be mapped to security-specific features and capabilities that must be built into software by developers.

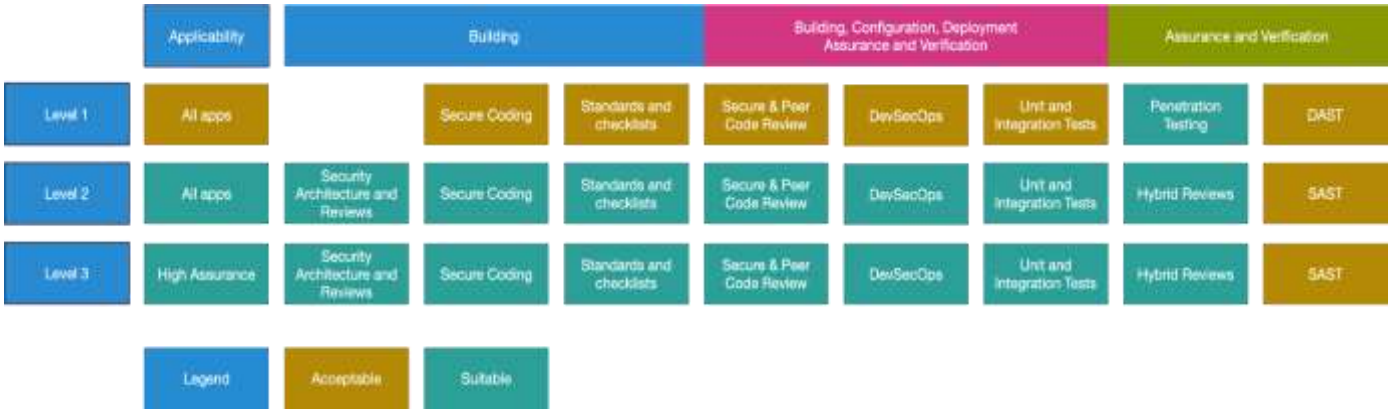


Figure 1 - OWASP Application Security Verification Standard 4.0 Levels

It is encouraged DAST (Dynamic Application Security Testing) and SAST (Static Application Security Testing) tools being used continuously by the build pipeline to find easy to find security issues that should never be present.

Automated tools and online scans are unable to complete more than half of the ASVS without human assistance. If comprehensive test automation for each build is required, then a combination of custom

unit and integration tests, along with build initiated online scans are used. Business logic flaws and access control testing is only possible using human assistance. These should be turned into unit and integration tests.

0.1.1 Level 1 – Automated – low criticality

An application achieves ASVS Level 1 if it adequately defends against application security vulnerabilities that are easy to discover and included in the OWASP Top 10 2017 and other similar checklists. **A Level 1 verification covers OWASP Top 10 - 2017 requirements from A1 to A9**. **A10 related requirements cannot be pen tested** and need from interviews with developers / architects, screenshots and further evidences. A10 compliance corresponds to *V7 Error handling and logging verification requirements* included in this document.

Level 1 is the bare minimum that all applications should strive for. It is also useful as a first step in a multi-phase effort or when applications do not store or handle sensitive data and therefore do not need the more rigorous controls of Level 2 or 3. Level 1 controls can be checked either automatically by tools or simply manually without access to source code. We consider Level 1 the minimum required for all applications.

Threats to the application will most likely be from attackers who are using simple and low effort techniques to identify easy-to-find and easy-to-exploit vulnerabilities. This is in contrast to a determined attacker who will spend focused energy to specifically target the application. If data processed by your application has high value, you would rarely want to stop at a Level 1 review.

Level 1 is the only level that is completely penetration testable using humans. All others require access to documentation, source code, configuration, and the people involved in the development process. However, even if L1 allows "black box" (no documentation and no source) testing to occur, it is not effective assurance and must stop.

0.1.2 Level 2 – Standard

An application achieves ASVS Level 2 (or Standard) if it adequately defends against most of the risks associated with software today.

Level 2 ensures that security controls are in place, effective, and used within the application. Level 2 is typically appropriate for applications that handle significant business-to-business transactions, including those that implement business-critical or sensitive functions, or process other sensitive assets as sensitive personal data, or applications where data integrity is a critical facet to protect the business.

Threats to Level 2 applications will typically be skilled and motivated attackers focusing on specific targets using tools and techniques that are highly practiced and effective at discovering and exploiting weaknesses within applications.

0.1.3 Level 3 – Business critical

ASVS Level 3 is the highest level of verification within the ASVS. This level is typically reserved for applications that require significant levels of security verification, such as those that may be found within areas of critical infrastructure, managing EU classified information, etc.

ASVS Level 3 might be required for applications that perform critical functions, where failure could significantly impact the organization's operations, and even its survivability. Example guidance on the application of ASVS Level 3 is provided below. An application achieves ASVS Level 3 (or Advanced) if it adequately defends against advanced application security vulnerabilities and also demonstrates principles of good security design.

An application at ASVS Level 3 requires more in-depth analysis or architecture, coding, and testing than all the other levels. A secure application is modularized in a meaningful way (to facilitate resiliency, scalability, and most of all, layers of security), and each module (separated by network connection and/or physical instance) takes care of its own security responsibilities (defence in depth), that need to be properly documented. Responsibilities include controls for ensuring confidentiality (e.g. encryption), integrity (e.g. transactions, input validation), availability, authentication (including between systems), non-repudiation, authorization, and auditing (logging).

0.2 The Role of Automated Security Testing Tools

The use of automated penetration testing tools is encouraged to provide as much coverage as possible.

It is not possible to fully complete ASVS verification using automated penetration testing tools alone. Whilst a large majority of requirements in L1 can be performed using automated tests, most requirements are not amenable to automated penetration testing.

Please note that automated tools are often manually tuned by experts and manual testers often leverage a wide variety of automated tools.

0.3 The Role of Penetration Testing

L1 is completely black box penetration testable without access to source code, documentation, or developers. Two logging items, which are required to comply with OWASP Top 10 2017 A10, will require interviews, screenshots or other evidence collection, just as they do in the OWASP Top 10 2017. However, testing without access to necessary information is not an ideal method of security verification, as it misses out on the possibility of reviewing the source, identifying threats and missing controls, and performing a far more thorough test in a shorter timeframe.

Where possible, access to developers, documentation, code, and access to a test application with non-production data, is required when performing a L2 or L3 Assessment. Penetration testing done at these levels requires this level of access, which is call "hybrid reviews" or "hybrid penetration tests".

1. V1: Architecture, Design and Threat Modelling Requirements

In this chapter, it is covered off the primary aspects of any security architecture: availability, confidentiality, processing integrity, non-repudiation, and privacy. Each of these security principles must be built in and be innate to all applications. It is critical to start with developer enablement with secure coding checklists, training, coding and testing, building, deployment, configuration, and operations, and finishing with follow up independent testing to assure that all the security controls are present and functional.

1.1 V1.1 Secure Software Development Lifecycle Requirements

#	Description	L1	L2	L3	CWE
1.1.1	Verify the use of a secure software development lifecycle that addresses security in all stages of development. (C1)		✓	✓	
1.1.2	Verify the use of threat modeling for every design change or sprint planning to identify threats, plan for countermeasures, facilitate appropriate risk responses, and guide security testing.		✓	✓	1053
1.1.3	Verify that all user stories and features contain functional security constraints, such as "As a user, I should be able to view and edit my profile. I should not be able to view or edit anyone else's profile"		✓	✓	1110
1.1.4	Verify documentation and justification of all the application's trust boundaries, components, and significant data flows.		✓	✓	1059
1.1.5	Verify definition and security analysis of the application's high-level architecture and all connected remote services. (C1)		✓	✓	1059
1.1.6	Verify implementation of centralized, simple (economy of design), vetted, secure, and reusable security controls to avoid duplicate, missing, ineffective, or insecure controls. (C10)		✓	✓	637
1.1.7	Verify availability of a secure coding checklist, security requirements, guideline, or policy to all developers and testers.		✓	✓	637

1.2 V1.2 Authentication Architectural Requirements

#	Description	L1	L2	L3	CWE
1.2.1	Verify the use of unique or special low-privilege operating system accounts for all application components, services, and servers. (C3)		✓	✓	250
1.2.2	Verify that communications between application components, including APIs, middleware and data layers, are authenticated. Components should have the least necessary privileges needed. (C3)		✓	✓	306
1.2.3	Verify that the application uses a single vetted authentication mechanism that is known to be secure, can be extended to include strong authentication, and has sufficient logging and monitoring to detect account abuse or breaches.		✓	✓	306
1.2.4	Verify that all authentication pathways and identity management APIs implement consistent authentication security control strength, such that there are no weaker alternatives per the risk of the application.		✓	✓	306

1.3 V1.3 Session Management Architectural Requirements

N/A

1.4 V1.4 Access Control Architectural Requirements

#	Description	L1	L2	L3	CWE
1.4.1	Verify that trusted enforcement points such as at access control gateways, servers, and serverless functions enforce access controls. Never enforce access controls on the client.		✓	✓	602
1.4.2	Verify that the chosen access control solution is flexible enough to meet the application's needs.		✓	✓	284
1.4.3	Verify enforcement of the principle of least privilege in functions, data files, URLs, controllers, services, and other resources. This implies protection against spoofing and elevation of privilege.		✓	✓	272
1.4.4	Verify the application uses a single and well-vetted access control mechanism for accessing protected data and resources. All requests must pass through this single mechanism to avoid copy and paste or insecure alternative paths. (C7)		✓	✓	284
1.4.5	Verify that attribute or feature-based access control is used whereby the code checks the user's authorization for a feature/data item rather than just their role. Permissions should still be allocated using roles. (C7)		✓	✓	275

1.5 V1.5 Input and Output Architectural Requirements

#	Description	L1	L2	L3	CWE
1.5.1	Verify that input and output requirements clearly define how to handle and process data based on type, content, and applicable laws, regulations, and other policy compliance.		✓	✓	1029
1.5.2	Verify that serialization is not used when communicating with untrusted clients. If this is not possible, ensure that adequate integrity controls (and possibly encryption if sensitive data is sent) are enforced to prevent deserialization attacks including object injection.		✓	✓	502
1.5.3	Verify that input validation is enforced on a trusted service layer. (C5)		✓	✓	602
1.5.4	Verify that output encoding occurs close to or by the interpreter for which it is intended. (C4)		✓	✓	116

1.6 V1.6 Cryptographic Architectural Requirements

#	Description	L1	L2	L3	CWE
1.6.1	Verify that there is an explicit policy for management of cryptographic keys and that a cryptographic key lifecycle follows a key management standard such as NIST SP 800-57.		✓	✓	320
1.6.2	Verify that consumers of cryptographic services protect key material and other secrets by using key vaults or API based alternatives.		✓	✓	320
1.6.3	Verify that all keys and passwords are replaceable and are part of a well-defined process to re-encrypt sensitive data.		✓	✓	320
1.6.4	Verify that symmetric keys, passwords, or API secrets generated by or shared with clients are used only in protecting low risk secrets, such as		✓	✓	320

#	Description	L1	L2	L3	CWE
	encrypting local storage, or temporary ephemeral uses such as parameter obfuscation. Sharing secrets with clients is clear-text equivalent and architecturally should be treated as such.				

1.7 V1.7 Errors, Logging and Auditing Architectural Requirements

#	Description	L1	L2	L3	CWE
1.7.1	Verify that a common logging format and approach is used across the system. (C9)		✓	✓	1009
1.7.2	Verify that logs are securely transmitted to a preferably remote system for analysis, detection, alerting, and escalation. (C9)		✓	✓	

1.8 V1.8 Data Protection and Privacy Architectural Requirements

#	Description	L1	L2	L3	CWE
1.8.1	Verify that all sensitive data is identified and classified into protection levels.		✓	✓	
1.8.2	Verify that all protection levels have an associated set of protection requirements, such as encryption requirements, integrity requirements, retention, privacy and other confidentiality requirements, and that these are applied in the architecture.		✓	✓	

1.9 V1.9 Communications Architectural Requirements

#	Description	L1	L2	L3	CWE
1.9.1	Verify the application encrypts communications between components, particularly when these components are in different containers, systems, sites, or cloud providers. (C3)		✓	✓	319
1.9.2	Verify that application components verify the authenticity of each side in a communication link to prevent person-in-the-middle attacks. For		✓	✓	295

example, application components should validate TLS certificates and chains.

1.10 V1.10 Malicious Software Architectural Requirements

#	Description	L1	L2	L3	CWE
1.10.1	Verify that a source code control system is in use, with procedures to ensure that check-ins are accompanied by issues or change tickets. The source code control system should have access control and identifiable users to allow traceability of any changes.		✓	✓	284

1.11 V1.11 Business Logic Architectural Requirements

#	Description	L1	L2	L3	CWE
1.11.1	Verify the definition and documentation of all application components in terms of the business or security functions they provide.		✓	✓	1059
1.11.2	Verify that all high-value business logic flows, including authentication, session management and access control, do not share unsynchronized state.		✓	✓	362
1.11.3	Verify that all high-value business logic flows, including authentication, session management and access control are thread safe and resistant to time-of-check and time-of-use race conditions.			✓	367

1.12 V1.12 Secure File Upload Architectural Requirements

#	Description	L1	L2	L3	CWE
1.12.1	Verify that user-uploaded files are stored outside of the web root.		✓	✓	552
1.12.2	Verify that user-uploaded files - if required to be displayed or downloaded from the application - are served by either octet stream downloads, or from an unrelated domain, such as a cloud file storage bucket. Implement a suitable content		✓	✓	646

security policy to reduce the risk from XSS vectors or other attacks from the uploaded file.

1.13 V1.13 API Architectural Requirements

This is a placeholder for future architectural requirements.

1.14 V1.14 Configuration Architectural Requirements

#	Description	L1	L2	L3	CWE
1.14.1	Verify the segregation of components of differing trust levels through well-defined security controls, firewall rules, API gateways, reverse proxies, cloud-based security groups, or similar mechanisms.		✓	✓	923
1.14.2	Verify that if deploying binaries to untrusted devices makes use of binary signatures, trusted connections, and verified endpoints.		✓	✓	494
1.14.3	Verify that the build pipeline warns of out-of-date or insecure components and takes appropriate actions.		✓	✓	1104
1.14.4	Verify that the build pipeline contains a build step to automatically build and verify the secure deployment of the application, particularly if the application infrastructure is software defined, such as cloud environment build scripts.		✓	✓	
1.14.5	Verify that application deployments adequately sandbox, containerize and/or isolate at the network level to delay and deter attackers from attacking other applications, especially when they are performing sensitive or dangerous actions such as deserialization. (C5)		✓	✓	265
1.14.6	Verify the application does not use unsupported, insecure, or deprecated client-side technologies such as NSAPI plugins, Flash, Shockwave, ActiveX, Silverlight, NACL, or client-side Java applets.		✓	✓	477

2. V2: Authentication Verification Requirements

2.1 V2.1 Password Security Requirements

This type of authenticator is considered "something you know".

#	Description	L1	L2	L3	CWE	NIST §
2.1.1	Verify that user set passwords are at least 12 characters in length. (C6)	✓	✓	✓	521	5.1.1.2
2.1.2	Verify that passwords 64 characters or longer are permitted. (C6)	✓	✓	✓	521	5.1.1.2
2.1.3	Verify that passwords can contain spaces and truncation is not performed. Consecutive multiple spaces MAY optionally be coalesced. (C6)	✓	✓	✓	521	5.1.1.2
2.1.4	Verify that Unicode characters are permitted in passwords. A single Unicode code point is considered a character, so 12 emoji or 64 kanji characters should be valid and permitted.	✓	✓	✓	521	5.1.1.2
2.1.5	Verify users can change their password.	✓	✓	✓	620	5.1.1.2
2.1.6	Verify that password change functionality requires the user's current and new password.	✓	✓	✓	620	5.1.1.2
2.1.7	Verify that passwords submitted during account registration, login, and password change are checked against a set of breached passwords either locally (such as the top 1,000 or 10,000 most common passwords which match the system's password policy) or using an external API. If using an API a zero knowledge proof or other mechanism should be used to ensure that the plain text password is not sent or used in verifying the breach status of the password. If the password is breached, the application must require the user to set a new non-breached password. (C6)	✓	✓	✓	521	5.1.1.2

#	Description	L1	L2	L3	CWE	NIST §
2.1.8	Verify that a password strength meter is provided to help users set a stronger password.	✓	✓	✓	521	5.1.1.2
2.1.9	Verify that there are no password composition rules limiting the type of characters permitted. There should be no requirement for upper or lower case or numbers or special characters. (C6)	✓	✓	✓	521	5.1.1.2
2.1.10	Verify that there are no periodic credential rotation or password history requirements.	✓	✓	✓	263	5.1.1.2
2.1.11	Verify that "paste" functionality, browser password helpers, and external password managers are permitted.	✓	✓	✓	521	5.1.1.2
2.1.12	Verify that the user can choose to either temporarily view the entire masked password, or temporarily view the last typed character of the password on platforms that do not have this as native functionality.	✓	✓	✓	521	5.1.1.2

2.2 V2.2 General Authenticator Requirements

#	Description	L1	L2	L3	CWE	NIST §
2.2.1	Verify that anti-automation controls are effective at mitigating breached credential testing, brute force, and account lockout attacks. Such controls include blocking the most common breached passwords, soft lockouts, rate limiting, CAPTCHA, ever increasing delays between attempts, IP address restrictions, or risk-based restrictions such as location, first login on a device, recent attempts to unlock the account, or similar. Verify that no more than 100 failed attempts per	✓	✓	✓	307	5.2.2 / 5.1.1.2 / 5.1.4.2 / 5.1.5.2

#	Description	L1	L2	L3	CWE	NIST §
	hour is possible on a single account.					
2.2.2	Verify that the use of weak authenticators (such as SMS and email) is limited to secondary verification and transaction approval and not as a replacement for more secure authentication methods. Verify that stronger methods are offered before weak methods, users are aware of the risks, or that proper measures are in place to limit the risks of account compromise.	✓	✓	✓	304	5.2.10
2.2.3	Verify that secure notifications are sent to users after updates to authentication details, such as credential resets, email or address changes, logging in from unknown or risky locations. The use of push notifications - rather than SMS or email - is preferred, but in the absence of push notifications, SMS or email is acceptable as long as no sensitive information is disclosed in the notification.	✓	✓	✓	620	
2.2.4	Verify impersonation resistance against phishing, such as the use of multi-factor authentication, cryptographic devices with intent (such as connected keys with a push to authenticate), or at higher AAL levels, client-side certificates.			✓	308	5.2.5
2.2.5	Verify that where a credential service provider (CSP) and the application verifying authentication are separated, mutually authenticated TLS is in place between the two endpoints.			✓	319	5.2.6
2.2.6	Verify replay resistance through the mandated use of OTP devices, cryptographic authenticators, or lookup codes.			✓	308	5.2.8
2.2.7	Verify intent to authenticate by requiring the entry of an OTP token or user-initiated action such			✓	308	5.2.9

#	Description	L1	L2	L3	CWE	NIST §
	as a button press on a FIDO hardware key.					

2.3 V2.3 Authenticator Lifecycle Requirements

Authenticators are passwords, soft tokens, hardware tokens, and biometric devices.

Note: Passwords should be checked for being breached.

#	Description	L1	L2	L3	CWE	NIST §
2.3.1	Verify system generated initial passwords or activation codes SHOULD be securely randomly generated, SHOULD be at least 6 characters long, and MAY contain letters and numbers, and expire after a short period of time. These initial secrets must not be permitted to become the long-term password.	✓	✓	✓	330	5.1.1.2 / A.3
2.3.2	Verify that enrolment and use of subscriber-provided authentication devices are supported, such as a U2F or FIDO tokens.		✓	✓	308	6.1.3
2.3.3	Verify that renewal instructions are sent with sufficient time to renew time bound authenticators.		✓	✓	287	6.1.4

2.4 V2.4 Credential Storage Requirements

Architects and developers should adhere to this section when building or refactoring code. This section can only be fully verified using source code review or through secure unit or integration tests. Penetration testing cannot identify any of these issues.

This section cannot be penetration tested, so controls are not marked as L1. However, this section is of the utmost importance to the security of credentials if they are stolen.

#	Description	L1	L2	L3	CWE	NIST §
2.4.1	Verify that passwords are stored in a form that is resistant to offline attacks. Passwords SHALL be salted and hashed using an approved one-way key derivation or password hashing function. Key		✓	✓	916	5.1.1.2

	derivation and password hashing functions take a password, a salt, and a cost factor as inputs when generating a password hash. (C6)					
2.4.2	Verify that the salt is at least 32 bits in length and be chosen arbitrarily to minimize salt value collisions among stored hashes. For each credential, a unique salt value and the resulting hash SHALL be stored. (C6)		✓	✓	916	5.1.1.2
2.4.3	Verify that if PBKDF2 is used, the iteration count SHOULD be as large as verification server performance will allow, typically at least 100,000 iterations. (C6)		✓	✓	916	5.1.1.2
2.4.4	Verify that if bcrypt is used, the work factor SHOULD be as large as verification server performance will allow, typically at least 13. (C6)		✓	✓	916	5.1.1.2
2.4.5	Verify that an additional iteration of a key derivation function is performed, using a salt value that is secret and known only to the verifier. Generate the salt value using an approved random bit generator and provide at least the minimum security strength specified in the latest revision of EMSA Policy (or SP 800-131A if not available). The secret salt value SHALL be stored separately from the hashed passwords (e.g., in a specialized device like a hardware security module).		✓	✓	916	5.1.1.2

2.5 V2.5 Credential Recovery Requirements

#	Description	L1	L2	L3	CWE	NIST §
2.5.1	Verify that a system generated initial activation or recovery secret is not sent in clear text to the user. (C6)	✓	✓	✓	640	5.1.1.2
2.5.2	Verify password hints or knowledge-based authentication	✓	✓	✓	640	5.1.1.2

#	Description	L1	L2	L3	CWE	NIST §
	(so-called "secret questions") are not present.					
2.5.3	Verify password credential recovery does not reveal the current password in any way. (C6)	✓	✓	✓	640	5.1.1.2
2.5.4	Verify shared or default accounts are not present (e.g. "root", "admin", or "sa").	✓	✓	✓	16	5.1.1.2 / A.3
2.5.5	Verify that if an authentication factor is changed or replaced, that the user is notified of this event.	✓	✓	✓	304	6.1.2.3
2.5.6	Verify forgotten password, and other recovery paths use a secure recovery mechanism, such as TOTP or other soft token, mobile push, or another offline recovery mechanism. (C6)	✓	✓	✓	640	5.1.1.2
2.5.7	Verify that if OTP or multi-factor authentication factors are lost, that evidence of identity proofing is performed at the same level as during enrollment.		✓	✓	308	6.1.2.3

2.6 V2.6 Look-up Secret Verifier Requirements

Look up secrets are pre-generated lists of secret codes, similar to Transaction Authorization Numbers (TAN), social media recovery codes, or a grid containing a set of random values. These are distributed securely to users. These lookup codes are used once, and once all used, the lookup secret list is discarded. This type of authenticator is considered "something you have".

#	Description	L1	L2	L3	CWE	NIST §
2.6.1	Verify that lookup secrets can be used only once.		✓	✓	308	5.1.2.2
2.6.2	Verify that lookup secrets have sufficient randomness (112 bits of entropy), or if less than 112 bits of entropy, salted with a unique and random 32-bit salt and hashed with an approved one-way hash.		✓	✓	330	5.1.2.2

2.6.3	Verify that lookup secrets are resistant to offline attacks, such as predictable values.		✓	✓	310	5.1.2.2

2.7 V2.7 Out of Band Verifier Requirements

Secure out of band authenticators are physical devices that can communicate with the verifier over a secure secondary channel. Examples include push notifications to mobile devices. This type of authenticator is considered "something you have".

#	Description	L1	L2	L3	CWE	NIST §
2.7.1	Verify that clear text out of band authenticators, such as SMS or PSTN, are not offered by default, and stronger alternatives such as push notifications are offered first.	✓	✓	✓	287	5.1.3.2
2.7.2	Verify that the out of band verifier expires out of band authentication requests, codes, or tokens after 10 minutes.	✓	✓	✓	287	5.1.3.2
2.7.3	Verify that the out of band verifier authentication requests, codes, or tokens are only usable once, and only for the original authentication request.	✓	✓	✓	287	5.1.3.2
2.7.4	Verify that the out of band authenticator and verifier communicates over a secure independent channel.	✓	✓	✓	523	5.1.3.2
2.7.5	Verify that the out of band verifier retains only a hashed version of the authentication code.		✓	✓	256	5.1.3.2
2.7.6	Verify that the initial authentication code is generated by a secure random number generator, containing at least 20 bits of entropy (typically a six digital random number is sufficient).		✓	✓	310	5.1.3.2

2.8 V2.8 Single or Multi Factor One Time Verifier Requirements

Single factor one-time passwords (OTPs) are physical or soft tokens that display a continually changing pseudo-random one time challenge.

#	Description	L1	L2	L3	CWE	NIST §
2.8.1	Verify that time-based OTPs have a defined lifetime before expiring.	✓	✓	✓	613	5.1.4.2 / 5.1.5.2
2.8.2	Verify that symmetric keys used to verify submitted OTPs are highly protected, such as by using a hardware security module or secure operating system based key storage.		✓	✓	320	5.1.4.2 / 5.1.5.2
2.8.3	Verify that approved cryptographic algorithms are used in the generation, seeding, and verification.		✓	✓	326	5.1.4.2 / 5.1.5.2
2.8.4	Verify that time-based OTP can be used only once within the validity period.		✓	✓	287	5.1.4.2 / 5.1.5.2
2.8.5	Verify that if a time-based multi factor OTP token is re-used during the validity period, it is logged and rejected with secure notifications being sent to the holder of the device.		✓	✓	287	5.1.5.2
2.8.6	Verify physical single factor OTP generator can be revoked in case of theft or other loss. Ensure that revocation is immediately effective across logged in sessions, regardless of location.		✓	✓	613	5.2.1
2.8.7	Verify that biometric authenticators are limited to use only as secondary factors in conjunction with either something you have and something you know.			✓	308	5.2.3

2.9 V2.9 Cryptographic Software and Devices Verifier Requirements

Cryptographic security keys are smart cards or FIDO keys, where the user has to plug in or pair the cryptographic device to the computer to complete authentication. Verifiers send a challenge nonce to the cryptographic devices or software, and the device or software calculates a response based upon a securely stored cryptographic key.

#	Description	L1	L2	L3	CWE	NIST §
2.9.1	Verify that cryptographic keys used in verification are stored securely and protected against disclosure, such as using a TPM or HSM, or an OS service that can use this secure storage.		✓	✓	320	5.1.7.2
2.9.2	Verify that the challenge nonce is at least 64 bits in length, and statistically unique or unique over the lifetime of the cryptographic device.		✓	✓	330	5.1.7.2
2.9.3	Verify that approved cryptographic algorithms are used in the generation, seeding, and verification.		✓	✓	327	5.1.7.2

2.10 V2.10 Service Authentication Requirements

This section is not penetration testable, so does not have any L1 requirements. However, if used in an architecture, coding or secure code review, please assume that software (just as Java Key Store) is the minimum requirement at L1. Clear text storage of secrets is not acceptable under any circumstances.

#	Description	L1	L2	L3	CWE	NIST §
2.10.1	Verify that integration secrets do not rely on unchanging passwords, such as API keys or shared privileged accounts.		OS assisted	HSM	287	5.1.1.1
2.10.2	Verify that if passwords are required, the credentials are not a default account.		OS assisted	HSM	255	5.1.1.1
2.10.3	Verify that passwords are stored with sufficient protection to prevent offline recovery attacks, including local system access.		OS assisted	HSM	522	5.1.1.1

2.10.4	Verify passwords, integrations with databases and third-party systems, seeds and internal secrets, and API keys are managed securely and not included in the source code or stored within source code repositories. Such storage SHOULD resist offline attacks. The use of a secure software key store (L1), hardware trusted platform module (TPM), or a hardware security module (L3) is recommended for password storage.	OS assisted	HSM	798	

3. V3: Session Management Verification Requirements

One of the core components of any web-based application or stateful API is the mechanism by which it controls and maintains the state for a user or device interacting with it. Session management changes a stateless protocol to stateful, which is critical for differentiating different users or devices.

To ensure that a verified application satisfies the following high-level session management requirements:

- Sessions are unique to each individual and cannot be guessed or shared.
- Sessions are invalidated when no longer required and timed out during periods of inactivity.

3.1 V3.1 Fundamental Session Management Requirements

#	Description	L1	L2	L3	CWE	NIST §
3.1.1	Verify the application never reveals session tokens in URL parameters or error messages.	✓	✓	✓	598	

3.2 V3.2 Session Binding Requirements

#	Description	L1	L2	L3	CWE	NIST §

3.2.1	Verify the application generates a new session token on user authentication. (C6)	✓	✓	✓	384	7.1
3.2.2	Verify that session tokens possess at least 64 bits of entropy. (C6)	✓	✓	✓	331	7.1
3.2.3	Verify the application only stores session tokens in the browser using secure methods such as appropriately secured cookies (see section 3.4) or HTML 5 session storage.	✓	✓	✓	539	7.1
3.2.4	Verify that session token are generated using approved cryptographic algorithms. (C6)		✓	✓	331	7.1

TLS or another secure transport channel is mandatory for session management. This is covered off in the Communications Security chapter.

3.3 V3.3 Session Logout and Timeout Requirements

#	Description	L1	L2	L3	CWE	NIST §
3.3.1	Verify that logout and expiration invalidate the session token, such that the back button or a downstream relying party does not resume an authenticated session, including across relying parties. (C6)	✓	✓	✓	613	7.1
3.3.2	If authenticators permit users to remain logged in, verify that re-authentication occurs periodically both when actively used or after an idle period (C6) – check EMSA Policy for periods.	✓	✓	✓	613	7.2
3.3.3	Verify that the application terminates all other active sessions after a successful password change, and that this is effective across the application, federated login (if present), and any relying parties.		✓	✓	613	
3.3.4	Verify that users are able to view and log out of any or all currently active sessions and devices.		✓	✓	613	7.1

3.4 V3.4 Cookie-based Session Management

#	Description	L1	L2	L3	CWE	NIST §
3.4.1	Verify that cookie-based session tokens have the 'Secure' attribute set. (C6)	✓	✓	✓	614	7.1.1
3.4.2	Verify that cookie-based session tokens have the 'HttpOnly' attribute set. (C6)	✓	✓	✓	1004	7.1.1
3.4.3	Verify that cookie-based session tokens utilize the 'SameSite' attribute to limit exposure to cross-site request forgery attacks. (C6)	✓	✓	✓	16	7.1.1
3.4.4	Verify that cookie-based session tokens use "__Host-" prefix (see references) to provide session cookie confidentiality.	✓	✓	✓	16	7.1.1
3.4.5	Verify that if the application is published under a domain name with other applications that set or use session cookies that might override or disclose the session cookies, set the path attribute in cookie-based session tokens using the most precise path possible. (C6)	✓	✓	✓	16	7.1.1

3.5 V3.5 Token-based Session Management

Token-based session management includes JWT, OAuth, SAML, and API keys. Of these, API keys are known to be weak and should not be used in new code.

#	Description	L1	L2	L3	CWE	NIST §
3.5.1	Verify the application does not treat OAuth and refresh tokens — on their own — as the presence of the subscriber and allows users to terminate trust relationships with linked applications.		✓	✓	290	7.1.2
3.5.2	Verify the application uses session tokens rather than static API secrets		✓	✓	798	

	and keys, except with legacy implementations.					
3.5.3	Verify that stateless session tokens use digital signatures, encryption, and other countermeasures to protect against tampering, enveloping, replay, null cipher, and key substitution attacks.		✓	✓	345	

3.6 V3.6 Re-authentication from a Federation or Assertion

This section relates to those writing relying party (RP) or credential service provider (CSP) code. If relying on code implementing these features, ensure that these issues are handled correctly.

#	Description	L1	L2	L3	CWE	NIST §
3.6.1	Verify that relying parties specify the maximum authentication time to CSPs and that CSPs re-authenticate the subscriber if they haven't used a session within that period.			✓	613	7.2.1
3.6.2	Verify that CSPs inform relying parties of the last authentication event, to allow RPs to determine if they need to re-authenticate the user.			✓	613	7.2.1

3.7 V3.7 Defenses Against Session Management Exploits

There are a small number of session management attacks, some related to the user experience (UX) of sessions. This section provides leading guidance on deterring, delaying and detecting session management attacks using code.

Description of the half-open Attack

In early 2018, several financial institutions were compromised using what the attackers called "half-open attacks". This term has stuck in the industry. The attackers struck multiple institutions with different proprietary code bases, and indeed it seems different code bases within the same institutions. The half-open attack is exploiting a design pattern flaw commonly found in many existing authentication, session management and access control systems.

Attackers start a half-open attack by attempting to lock, reset, or recover a credential. A popular session management design pattern re-uses user profile session objects/models between unauthenticated, half-authenticated (password resets, forgot username), and fully authenticated code. This design pattern populates a valid session object or token containing the victim's profile, including password hashes and roles. If access control checks in controllers or routers does not correctly verify that the user is fully logged in, the attacker will be able to act as the user. Attacks could include changing the user's password to a known value, update the email address to perform a valid password reset, disable multi-factor authentication or enrol a new MFA device, reveal or change API keys, and so on.

#	Description	L1	L2	L3	CWE	NIST §
3.7.1	Verify the application ensures a valid login session or requires re-authentication or secondary verification before allowing any sensitive transactions or account modifications.	✓	✓	✓	778	

4. V4: Access Control Verification Requirements

Authorization is the concept of allowing access to resources only to those permitted to use them. Ensure that a verified application satisfies the following high-level requirements:

- Persons accessing resources hold valid credentials to do so.
- Users are associated with a well-defined set of roles and privileges.
- Role and permission metadata is protected from replay or tampering.

4.1 V4.1 General Access Control Design

#	Description	L1	L2	L3	CWE
4.1.1	Verify that the application enforces access control rules on a trusted service layer, especially if client-side access control is present and could be bypassed.	✓	✓	✓	602
4.1.2	Verify that all user and data attributes and policy information used by access controls cannot be manipulated by end users unless specifically authorized.	✓	✓	✓	639
4.1.3	Verify that the principle of least privilege exists - users should only be able to access functions, data files, URLs, controllers, services, and other resources, for which they possess specific authorization. This implies protection against spoofing and elevation of privilege. (C7)	✓	✓	✓	285
4.1.4	Verify that the principle of deny by default exists whereby new users/roles start with minimal or no permissions and users/roles do not receive access to new features until access is explicitly assigned. (C7)	✓	✓	✓	276

4.1.5	Verify that access controls fail securely including when an exception occurs. (C10)	✓	✓	✓	285

4.2 V4.2 Operation Level Access Control

#	Description	L1	L2	L3	CWE
4.2.1	Verify that sensitive data and APIs are protected against direct object attacks targeting creation, reading, updating and deletion of records, such as creating or updating someone else's record, viewing everyone's records, or deleting all records.	✓	✓	✓	639
4.2.2	Verify that the application or framework enforces a strong anti-CSRF mechanism to protect authenticated functionality, and effective anti-automation or anti-CSRF protects unauthenticated functionality.	✓	✓	✓	352

4.3 V4.3 Other Access Control Considerations

#	Description	L1	L2	L3	CWE
4.3.1	Verify administrative interfaces use appropriate multi-factor authentication to prevent unauthorized use.	✓	✓	✓	419
4.3.2	Verify that directory browsing is disabled unless deliberately desired. Additionally, applications should not allow discovery or disclosure of file or directory metadata, such as Thumbs.db, .DS_Store, .git or .svn folders.	✓	✓	✓	548
4.3.3	Verify the application has additional authorization (such as step up or adaptive authentication) for lower value systems, and / or segregation of duties for high value applications to enforce anti-fraud controls as per the risk of application and past fraud.		✓	✓	732

5. V5: Validation, Sanitization and Encoding Verification Requirements

The most common web application security weakness is the failure to properly validate input coming from the client or the environment before directly using it without any output encoding. This weakness leads to almost all of the significant vulnerabilities in web applications, such as Cross-Site Scripting (XSS), SQL injection, interpreter injection, locale/Unicode attacks, file system attacks, and buffer overflows.

To ensure that a verified application satisfies the following high-level requirements:

- Input validation and output encoding architecture have an agreed pipeline to prevent injection attacks.
- Input data is strongly typed, validated, range or length checked, or at worst, sanitized or filtered.
- Output data is encoded or escaped as per the context of the data as close to the interpreter as possible.

With modern web application architecture, it is difficult to provide robust input validation in certain scenarios, so the use of safer API such as parameterized queries, auto-escaping templating frameworks, or carefully chosen output encoding is critical to the security of the application.

5.1 V5.1 Input Validation Requirements

Properly implemented input validation controls, using positive whitelisting and strong data typing, can eliminate more than 90% of all injection attacks. Length and range checks can reduce this further. Building in secure input validation is required during application architecture, design, coding, and unit and integration testing. Although many of these items cannot be found in penetration tests, the results of not implementing them are usually found in V5.3 - Output encoding and Injection Prevention Requirements. Developers and secure code reviewers are required to treat this section as for L1.

#	Description	L1	L2	L3	CWE
5.1.1	Verify that the application has defenses against HTTP parameter pollution attacks, particularly if the application framework makes no distinction about the source of request parameters (GET, POST, cookies, headers, or environment variables).	✓	✓	✓	235
5.1.2	Verify that frameworks protect against mass parameter assignment attacks, or that the application has countermeasures to protect against unsafe parameter assignment, such as marking fields private or similar. (C5)	✓	✓	✓	915
5.1.3	Verify that all input (HTML form fields, REST requests, URL parameters, HTTP headers,	✓	✓	✓	20

	cookies, batch files, RSS feeds, etc) is validated using positive validation (whitelisting). (C5)				
5.1.4	Verify that structured data is strongly typed and validated against a defined schema including allowed characters, length and pattern (e.g. credit card numbers or telephone, or validating that two related fields are reasonable, such as checking that suburb and zip/postcode match). (C5)	✓	✓	✓	20
5.1.5	Verify that URL redirects and forwards only allow whitelisted destinations, or show a warning when redirecting to potentially untrusted content.	✓	✓	✓	601

5.2 V5.2 Sanitization and Sandboxing Requirements

#	Description	L1	L2	L3	CWE
5.2.1	Verify that all untrusted HTML input from WYSIWYG editors or similar is properly sanitized with an HTML sanitizer library or framework feature. (C5)	✓	✓	✓	116
5.2.2	Verify that unstructured data is sanitized to enforce safety measures such as allowed characters and length.	✓	✓	✓	138
5.2.3	Verify that the application sanitizes user input before passing to mail systems to protect against SMTP or IMAP injection.	✓	✓	✓	147
5.2.4	Verify that the application avoids the use of eval() or other dynamic code execution features. Where there is no alternative, any user input being included must be sanitized or sandboxed before being executed.	✓	✓	✓	95
5.2.5	Verify that the application protects against template injection attacks by ensuring that any user input being included is sanitized or sandboxed.	✓	✓	✓	94
5.2.6	Verify that the application protects against SSRF attacks, by validating or sanitizing untrusted data or HTTP file metadata, such as filenames and URL input fields, use whitelisting of protocols, domains, paths and ports.	✓	✓	✓	918

5.2.7	Verify that the application sanitizes, disables, or sandboxes user-supplied SVG scriptable content, especially as they relate to XSS resulting from inline scripts, and foreignObject.	✓	✓	✓	159
5.2.8	Verify that the application sanitizes, disables, or sandboxes user-supplied scriptable or expression template language content, such as Markdown, CSS or XSL stylesheets, BBCode, or similar.	✓	✓	✓	94

5.3 V5.3 Output encoding and Injection Prevention Requirements

Output encoding close or adjacent to the interpreter in use is critical to the security of any application. Typically, output encoding is not persisted, but used to render the output safe in the appropriate output context for immediate use. Failing to output encode will result in an insecure, injectable, and unsafe application.

#	Description	L1	L2	L3	CWE
5.3.1	Verify that output encoding is relevant for the interpreter and context required. For example, use encoders specifically for HTML values, HTML attributes, JavaScript, URL Parameters, HTTP headers, SMTP, and others as the context requires, especially from untrusted inputs (e.g. names with Unicode or apostrophes, such as ねこ or O'Hara). (C4)	✓	✓	✓	116
5.3.2	Verify that output encoding preserves the user's chosen character set and locale, such that any Unicode character point is valid and safely handled. (C4)	✓	✓	✓	176
5.3.3	Verify that context-aware, preferably automated - or at worst, manual - output escaping protects against reflected, stored, and DOM based XSS. (C4)	✓	✓	✓	79
5.3.4	Verify that data selection or database queries (e.g. SQL, HQL, ORM, NoSQL) use parameterized queries, ORMs, entity frameworks, or are otherwise protected from database injection attacks. (C3)	✓	✓	✓	89
5.3.5	Verify that where parameterized or safer mechanisms are not present, context-specific output encoding is used to protect against injection attacks, such as the use of SQL escaping to protect against SQL injection. (C3, C4)	✓	✓	✓	89

#	Description	L1	L2	L3	CWE
5.3.6	Verify that the application projects against JavaScript or JSON injection attacks, including for eval attacks, remote JavaScript includes, CSP bypasses, DOM XSS, and JavaScript expression evaluation. (C4)	✓	✓	✓	830
5.3.7	Verify that the application protects against LDAP Injection vulnerabilities, or that specific security controls to prevent LDAP Injection have been implemented. (C4)	✓	✓	✓	943
5.3.8	Verify that the application protects against OS command injection and that operating system calls use parameterized OS queries or use contextual command line output encoding. (C4)	✓	✓	✓	78
5.3.9	Verify that the application protects against Local File Inclusion (LFI) or Remote File Inclusion (RFI) attacks.	✓	✓	✓	829
5.3.10	Verify that the application protects against XPath injection or XML injection attacks. (C4)	✓	✓	✓	643

Note: Using parameterized queries or escaping SQL is not always sufficient; table and column names, ORDER BY and so on, cannot be escaped. The inclusion of escaped user-supplied data in these fields results in failed queries or SQL injection.

Note: The SVG format explicitly allows ECMA script in almost all contexts, so it may not be possible to block all SVG XSS vectors completely. If SVG upload is required, recommendation is to either serving these uploaded files as text/plain or using a separate user supplied content domain to prevent successful XSS from taking over the application.

5.4 V5.4 Memory, String, and Unmanaged Code Requirements

The following requirements will only apply when the application uses a systems language or unmanaged code.

#	Description	L1	L2	L3	CWE
5.4.1	Verify that the application uses memory-safe string, safer memory copy and pointer arithmetic to detect or prevent stack, buffer, or heap overflows.		✓	✓	120
5.4.2	Verify that format strings do not take potentially hostile input, and are constant.		✓	✓	134

5.4.3	Verify that sign, range, and input validation techniques are used to prevent integer overflows.		✓	✓	190

5.5 V5.5 Deserialization Prevention Requirements

#	Description	L1	L2	L3	CWE
5.5.1	Verify that serialized objects use integrity checks or are encrypted to prevent hostile object creation or data tampering. (C5)	✓	✓	✓	502
5.5.2	Verify that the application correctly restricts XML parsers to only use the most restrictive configuration possible and to ensure that unsafe features such as resolving external entities are disabled to prevent XXE.	✓	✓	✓	611
5.5.3	Verify that deserialization of untrusted data is avoided or is protected in both custom code and third-party libraries (such as JSON, XML and YAML parsers).	✓	✓	✓	502
5.5.4	Verify that when parsing JSON in browsers or JavaScript-based backends, JSON.parse is used to parse the JSON document. Do not use eval() to parse JSON.	✓	✓	✓	95

6. V6: Stored Cryptography Verification Requirements

To ensure that a verified application satisfies the following high-level requirements:

- All cryptographic modules fail in a secure manner and that errors are handled correctly.
- A suitable random number generator is used.
- Access to keys is securely managed.

6.1 V6.1 Data Classification

#	Description	L1	L2	L3	CWE
6.1.1	Verify that regulated private data is stored encrypted while at rest, such as personally identifiable information (PII), sensitive personal		✓	✓	311

	information, or data assessed likely to be subject to EU's GDPR.				
6.1.2	Verify that regulated health data is stored encrypted while at rest, such as medical records, medical device details, or de-anonymized research records.		✓	✓	311
6.1.3	Verify that regulated financial data is stored encrypted while at rest, such as financial accounts, defaults or credit history, tax records, pay history, beneficiaries, or de-anonymized market or research records.		✓	✓	311

6.2 V6.2 Algorithms

Although this section is not easily penetration tested, developers shall consider this entire section as mandatory even though L1 is missing from most of the items.

#	Description	L1	L2	L3	CWE
6.2.1	Verify that all cryptographic modules fail securely, and errors are handled in a way that does not enable Padding Oracle attacks.	✓	✓	✓	310
6.2.2	Verify that industry proven or government approved cryptographic algorithms, modes, and libraries are used, instead of custom coded cryptography. (C8)	✓	✓	✓	327
6.2.3	Verify that encryption initialization vector, cipher configuration, and block modes are configured securely using the latest advice.	✓	✓	✓	326
6.2.4	Verify that random number, encryption or hashing algorithms, key lengths, rounds, ciphers or modes, can be reconfigured, upgraded, or swapped at any time, to protect against cryptographic breaks. (C8)	✓	✓	✓	326
6.2.5	Verify that known insecure block modes (i.e. ECB, etc.), padding modes (i.e. PKCS#1 v1.5, etc.), ciphers with small block sizes (i.e. Triple-DES, Blowfish, etc.), and weak hashing algorithms (i.e. MD5, SHA1, etc.) are not used unless required for backwards compatibility.	✓	✓	✓	326
6.2.6	Verify that nonces, initialization vectors, and other single use numbers must not be used		✓	✓	326

#	Description	L1	L2	L3	CWE
	more than once with a given encryption key. The method of generation must be appropriate for the algorithm being used.				
6.2.7	Verify that encrypted data is authenticated via signatures, authenticated cipher modes, or HMAC to ensure that ciphertext is not altered by an unauthorized party.			✓	326
6.2.8	Verify that all cryptographic operations are constant-time, with no 'short-circuit' operations in comparisons, calculations, or returns, to avoid leaking information.			✓	385

6.3 V6.3 Random Values

#	Description	L1	L2	L3	CWE
6.3.1	Verify that all random numbers, random file names, random GUIDs, and random strings are generated using the cryptographic module's approved cryptographically secure random number generator when these random values are intended to be not guessable by an attacker.		✓	✓	338
6.3.2	Verify that random GUIDs are created using the GUID v4 algorithm, and a cryptographically-secure pseudo-random number generator (CSPRNG). GUIDs created using other pseudo-random number generators may be predictable.		✓	✓	338
6.3.3	Verify that random numbers are created with proper entropy even when the application is under heavy load, or that the application degrades gracefully in such circumstances.			✓	338

6.4 V6.4 Secret Management (password management)

#	Description	L1	L2	L3	CWE
6.4.1	Verify that a secrets management solution such as a key vault is used to securely create, store, control access to and destroy secrets. (C8)	✓	✓	✓	798

6.4.2	Verify that key material is not exposed to the application but instead uses an isolated security module like a vault for cryptographic operations. (C8)	✓	✓	✓	320

7. V7: Error Handling and Logging Verification Requirements

The primary objective of error handling and logging is to provide useful information for the user, administrators, and incident response teams.

7.1 V7.1 Log Content Requirements

Logging sensitive information is dangerous they need to be encrypted, become subject to retention policies, and must be disclosed in security audits. Only necessary information shall be kept in logs, and certainly no credentials (including session tokens), sensitive or personally identifiable information.

V7.1 covers OWASP Top 10 2017:A10. As 2017:A10 and this section are not penetration testable, it's important for:

- Developers to ensure full compliance with this section, as if all items were marked as L1
- Penetration testers to validate full compliance of all items in V7.1 via interview, screenshots, or assertion

#	Description	L1	L2	L3	CWE
7.1.1	Verify that the application does not log credentials or payment details. Session tokens should only be stored in logs in an irreversible, hashed form. (C9, C10)	✓	✓	✓	532
7.1.2	Verify that the application does not log other sensitive data as defined under local privacy laws or relevant security policy. (C9)	✓	✓	✓	532
7.1.3	Verify that the application logs security relevant events including successful and failed authentication events, access control failures, deserialization failures and input validation failures. (C5, C7)		✓	✓	778
7.1.4	Verify that each log event includes necessary information that would allow for a detailed		✓	✓	778

investigation of the timeline when an event happens. (C9)

7.2 V7.2 Log Processing Requirements

V7.2 covers OWASP Top 10 2017:A10. As 2017:A10 and this section are not penetration testable, it's important for:

- Developers to ensure full compliance with this section, as if all items were marked as L1
- Penetration testers to validate full compliance of all items in V7.2 via interview, screenshots, or assertion

#	Description	L1	L2	L3	CWE
7.2.1	Verify that all authentication decisions are logged, without storing sensitive session identifiers or passwords. This should include requests with relevant metadata needed for security investigations.		✓	✓	778
7.2.2	Verify that all access control decisions can be logged and all failed decisions are logged. This should include requests with relevant metadata needed for security investigations.		✓	✓	285

7.3 V7.3 Log Protection Requirements

Logs that can be trivially modified or deleted are useless for investigations and prosecutions. Disclosure of logs can expose inner details about the application or the data it contains. Care must be taken when protecting logs from unauthorized disclosure, modification or deletion.

#	Description	L1	L2	L3	CWE
7.3.1	Verify that the application appropriately encodes user-supplied data to prevent log injection. (C9)		✓	✓	117
7.3.2	Verify that all events are protected from injection when viewed in log viewing software. (C9)		✓	✓	117
7.3.3	Verify that security logs are protected from unauthorized access and modification. (C9)		✓	✓	200
7.3.4	Verify that time sources are synchronized to the correct time and time zone. Strongly consider logging only in UTC if systems are global to assist with post-incident forensic analysis. (C9)		✓	✓	

Note: Log encoding (7.3.1) is difficult to test and review using automated dynamic tools and penetration tests, but architects, developers, and source code reviewers should consider it an L1 requirement.

7.4 V7.4 Error Handling

The purpose of error handling is to allow the application to provide security relevant events for monitoring, triage and escalation. The purpose is not to create logs. When logging security related events, ensure that there is a purpose to the log, and that it can be distinguished by SIEM or analysis software.

#	Description	L1	L2	L3	CWE
7.4.1	Verify that a generic message is shown when an unexpected or security sensitive error occurs, potentially with a unique ID which support personnel can use to investigate. (C10)	✓	✓	✓	210
7.4.2	Verify that exception handling (or a functional equivalent) is used across the codebase to account for expected and unexpected error conditions. (C10)		✓	✓	544
7.4.3	Verify that a "last resort" error handler is defined which will catch all unhandled exceptions. (C10)		✓	✓	460

8. V8: Data Protection Verification Requirements

There are three key elements to sound data protection: Confidentiality, Integrity and Availability (CIA). This standard assumes that data protection is enforced on a trusted system, such as a server, which has been hardened and has sufficient protections.

Applications have to assume that all user devices are compromised in some way. Where an application transmits or stores sensitive information on insecure devices, such as shared computers, phones and tablets, the application is responsible for ensuring data stored on these devices is encrypted and cannot be easily illicitly obtained, altered or disclosed.

Ensure that a verified application satisfies the following high level data protection requirements:

- Confidentiality: Data should be protected from unauthorized observation or disclosure both in transit and when stored.
- Integrity: Data should be protected from being maliciously created, altered or deleted by unauthorized attackers.
- Availability: Data should be available to authorized users as required.

8.1 V8.1 General Data Protection

#	Description	L1	L2	L3	CWE
8.1.1	Verify the application protects sensitive data from being cached in server components such as load balancers and application caches.		✓	✓	524
8.1.2	Verify that all cached or temporary copies of sensitive data stored on the server are protected from unauthorized access or purged/invalidated after the authorized user accesses the sensitive data.		✓	✓	524
8.1.3	Verify the application minimizes the number of parameters in a request, such as hidden fields, Ajax variables, cookies and header values.		✓	✓	233
8.1.4	Verify the application can detect and alert on abnormal numbers of requests, such as by IP, user, total per hour or day, or whatever makes sense for the application.		✓	✓	770
8.1.5	Verify that regular backups of important data are performed and that test restoration of data is performed.			✓	19
8.1.6	Verify that backups are stored securely to prevent data from being stolen or corrupted.			✓	19

8.2 V8.2 Client-side Data Protection

#	Description	L1	L2	L3	CWE
8.2.1	Verify the application sets sufficient anti-caching headers so that sensitive data is not cached in modern browsers.	✓	✓	✓	525
8.2.2	Verify that data stored in client side storage (such as HTML5 local storage, session storage, IndexedDB, regular cookies or Flash cookies) does not contain sensitive data or PII.	✓	✓	✓	922
8.2.3	Verify that authenticated data is cleared from client storage, such as the browser DOM, after the client or session is terminated.	✓	✓	✓	922

8.3 V8.3 Sensitive Private Data

This section helps protect sensitive data from being created, read, updated, or deleted without authorization, particularly in bulk quantities.

Compliance with this section implies compliance with V4 Access Control, and in particular V4.2. For example, to protect against unauthorized updates or disclosure of sensitive personal information requires adherence to V4.2.1. Please comply with this section and V4 for full coverage.

Note: Privacy regulations and laws, such as the Australian Privacy Principles APP-11 or GDPR, directly affect how applications must approach the implementation of storage, use, and transmission of sensitive personal information. This ranges from severe penalties to simple advice. Please consult your local laws and regulations, and consult a qualified privacy specialist or lawyer as required.

#	Description	L1	L2	L3	CWE
8.3.1	Verify that sensitive data is sent to the server in the HTTP message body or headers, and that query string parameters from any HTTP verb do not contain sensitive data.	✓	✓	✓	319
8.3.2	Verify that users have a method to remove or export their data on demand.	✓	✓	✓	212
8.3.3	Verify that users are provided clear language regarding collection and use of supplied personal information and that users have provided opt-in consent for the use of that data before it is used in any way.	✓	✓	✓	285
8.3.4	Verify that all sensitive data created and processed by the application has been identified,	✓	✓	✓	200

	and ensure that a policy is in place on how to deal with sensitive data. (C8)			
8.3.5	Verify accessing sensitive data is audited (without logging the sensitive data itself), if the data is collected under relevant data protection directives or where logging of access is required.	✓	✓	532
8.3.6	Verify that sensitive information contained in memory is overwritten as soon as it is no longer required to mitigate memory dumping attacks, using zeroes or random data.	✓	✓	226
8.3.7	Verify that sensitive or private information that is required to be encrypted, is encrypted using approved algorithms that provide both confidentiality and integrity. (C8)	✓	✓	327
8.3.8	Verify that sensitive personal information is subject to data retention classification, such that old or out of date data is deleted automatically, on a schedule, or as the situation requires.	✓	✓	285

When considering data protection, a primary consideration should be around bulk extraction or modification or excessive usage. For example, many social media systems only allow users to add 100 new friends per day, but which system these requests came from is not important. A banking platform might wish to block more than 5 transactions per hour transferring more than 1000 euro of funds to external institutions. Each system's requirements are likely to be very different, so deciding on "abnormal" must consider the threat model and business risk. Important criteria are the ability to detect, deter, or preferably block such abnormal bulk actions.

9. V9: Communications Verification Requirements

Ensure that a verified application satisfies the following high level requirements:

- TLS or strong encryption is always used, regardless of the sensitivity of the data being transmitted
- The most recent, leading configuration advice is used to enable and order preferred algorithms and ciphers
- Weak or soon to be deprecated algorithms and ciphers are ordered as a last resort
- Deprecated or known insecure algorithms and ciphers are disabled.

Leading industry advice on secure TLS configuration changes frequently, often due to catastrophic breaks in existing algorithms and ciphers. Always use the most recent versions of TLS configuration review tools (such as SSLyze or other TLS scanners) to configure the preferred order and algorithm selection. Configuration should be periodically checked to ensure that secure communications configuration is always present and effective.

9.1 V9.1 Communications Security Requirements

All client communications should only take place over encrypted communication paths. In particular, the use of TLS 1.2 or later is essentially all but required by modern browsers and search engines. Configuration should be regularly reviewed using online tools to ensure that the latest leading practices are in place.

#	Description	L1	L2	L3	CWE
9.1.1	Verify that secured TLS is used for all client connectivity, and does not fall back to insecure or unencrypted protocols. (C8)	✓	✓	✓	319
9.1.2	Verify using online or up to date TLS testing tools that only strong algorithms, ciphers, and protocols are enabled, with the strongest algorithms and ciphers set as preferred.	✓	✓	✓	326
9.1.3	Verify that old versions of SSL and TLS protocols, algorithms, ciphers, and configuration are disabled, such as SSLv2, SSLv3, or TLS 1.0 and TLS 1.1. The latest version of TLS should be the preferred cipher suite.	✓	✓	✓	326

9.2 V9.2 Server Communications Security Requirements

Server communications are more than just HTTP. Secure connections to and from other systems, such as monitoring systems, management tools, remote access and ssh, middleware, database, mainframes, partner or external source systems — must be in place. All of these must be encrypted to prevent "hard on the outside, trivially easy to intercept on the inside".

#	Description	L1	L2	L3	CWE
9.2.1	Verify that connections to and from the server use trusted TLS certificates. Where internally generated or self-signed certificates are used, the server must be configured to only trust specific internal CAs and specific self-signed certificates. All others should be rejected.		✓	✓	295
9.2.2	Verify that encrypted communications such as TLS is used for all inbound and outbound connections, including for management ports, monitoring, authentication, API, or web service calls, database, cloud, serverless, mainframe, external, and partner connections. The server must not fall back to insecure or unencrypted protocols.		✓	✓	319
9.2.3	Verify that all encrypted connections to external systems that involve sensitive information or functions are authenticated.		✓	✓	287
9.2.4	Verify that proper certification revocation, such as Online Certificate Status Protocol (OCSP) Stapling, is enabled and configured.		✓	✓	299
9.2.5	Verify that backend TLS connection failures are logged.			✓	544

10. V10: Malicious Code Verification Requirements

Ensure that code satisfies the following high level requirements:

- Malicious activity is handled securely and properly to not affect the rest of the application.
- Does not have time bombs or other time-based attacks.
- Does not "phone home" to malicious or unauthorized destinations.
- Does not have back doors, Easter eggs, salami attacks, rootkits, or unauthorized code that can be controlled by an attacker.

Finding malicious code is proof of the negative, which is impossible to completely validate. Best efforts should be undertaken to ensure that the code has no inherent malicious code or unwanted functionality.

10.1 V10.1 Code Integrity Controls

The best defense against malicious code is "trust, but verify". Introducing unauthorized or malicious code into code is often a criminal offence in many jurisdictions. Policies and procedures should make sanctions regarding malicious code clear.

Lead developers should regularly review code check-ins, particularly those that might access time, I/O, or network functions.

#	Description	L1	L2	L3	CWE
10.1.1	Verify that a code analysis tool is in use that can detect potentially malicious code, such as time functions, unsafe file operations and network connections.			✓	749

10.2 V10.2 Malicious Code Search

Malicious code is extremely rare and is difficult to detect. Manual line by line code review can assist looking for logic bombs, but even the most experienced code reviewer will struggle to find malicious code even if they know it exists.

Complying with this section is not possible without complete access to source code, including third-party libraries.

#	Description	L1	L2	L3	CWE
10.2.1	Verify that the application source code and third party libraries do not contain unauthorized phone home or data collection capabilities. Where such functionality exists, obtain the user's permission for it to operate before collecting any data.		✓	✓	359
10.2.2	Verify that the application does not ask for unnecessary or excessive permissions to privacy related features or sensors, such as contacts, cameras, microphones, or location.		✓	✓	272
10.2.3	Verify that the application source code and third party libraries do not contain back doors, such as hard-coded or additional undocumented accounts or keys, code obfuscation, undocumented binary blobs, rootkits, or anti-debugging, insecure debugging features, or otherwise out of date, insecure, or hidden functionality that could be used maliciously if discovered.			✓	507
10.2.4	Verify that the application source code and third party libraries does not contain time bombs by searching for date and time related functions.			✓	511
10.2.5	Verify that the application source code and third party libraries does not contain malicious code, such as salami attacks, logic bypasses, or logic bombs.			✓	511
10.2.6	Verify that the application source code and third party libraries do not contain Easter eggs or any other potentially unwanted functionality.			✓	507

10.3 V10.3 Deployed Application Integrity Controls

Once an application is deployed, malicious code can still be inserted. Applications need to protect themselves against common attacks, such as executing unsigned code from untrusted sources and sub-domain takeovers.

Complying with this section is likely to be operational and continuous.

#	Description	L1	L2	L3	CWE
10.3.1	Verify that if the application has a client or server auto-update feature, updates should be obtained over secure channels and digitally signed. The update code must validate the digital signature of the update before installing or executing the update.	✓	✓	✓	16
10.3.2	Verify that the application employs integrity protections, such as code signing or sub-resource integrity. The application must not load or execute code from untrusted sources, such as loading includes, modules, plugins, code, or libraries from untrusted sources or the Internet.	✓	✓	✓	353
10.3.3	Verify that the application has protection from sub-domain takeovers if the application relies upon DNS entries or DNS sub-domains, such as expired domain names, out of date DNS pointers or CNAMEs, expired projects at public source code repos, or transient cloud APIs, serverless functions, or storage buckets (*autogen-bucket-id*.cloud.example.com) or similar. Protections can include ensuring that DNS names used by applications are regularly checked for expiry or change.	✓	✓	✓	350

11. V11: Business Logic Verification Requirements

Ensure that a verified application satisfies the following high-level requirements:

- The business logic flow is sequential, processed in order, and cannot be bypassed.
- Business logic includes limits to detect and prevent automated attacks, such as continuous small funds transfers, or adding a million friends one at a time, and so on.
- High value business logic flows have considered abuse cases and malicious actors, and have protections against spoofing, tampering, repudiation, information disclosure, and elevation of privilege attacks.

11.1 V11.1 Business Logic Security Requirements

Business logic security is individual to every application. Business logic security must be designed in to protect against likely external threats - it cannot be added using web application firewalls or secure communications. The use of threat modelling during design is requested, for example using the OWASP Cornucopia or similar tools.

#	Description	L1	L2	L3	CWE
11.1.1	Verify the application will only process business logic flows for the same user in sequential step order and without skipping steps.	✓	✓	✓	841
11.1.2	Verify the application will only process business logic flows with all steps being processed in realistic human time, i.e. transactions are not submitted too quickly.	✓	✓	✓	779
11.1.3	Verify the application has appropriate limits for specific business actions or transactions which are correctly enforced on a per user basis.	✓	✓	✓	770
11.1.4	Verify the application has sufficient anti-automation controls to detect and protect against data exfiltration, excessive business logic requests, excessive file uploads or denial of service attacks.	✓	✓	✓	770
11.1.5	Verify the application has business logic limits or validation to protect against likely business risks or threats, identified using threat modelling or similar methodologies.	✓	✓	✓	841
11.1.6	Verify the application does not suffer from "time of check to time of use" (TOCTOU) issues or other race conditions for sensitive operations.		✓	✓	367

11.1.7	Verify the application monitors for unusual events or activity from a business logic perspective. For example, attempts to perform actions out of order or actions which a normal user would never attempt. (C9)		✓	✓	754
11.1.8	Verify the application has configurable alerting when automated attacks or unusual activity is detected.		✓	✓	390

12. V12: File and Resources Verification Requirements

To Ensure that a verified application satisfies the following high-level requirements:

- Untrusted file data should be handled accordingly and in a secure manner.
- Untrusted file data obtained from untrusted sources are stored outside the web root and with limited permissions.

12.1 V12.1 File Upload Requirements

Although zip bombs are eminently testable using penetration testing techniques, they are considered L2 and above to encourage design and development consideration with careful manual testing, and to avoid automated or unskilled manual penetration testing of a denial of service condition.

#	Description	L1	L2	L3	CWE
12.1.1	Verify that the application will not accept large files that could fill up storage or cause a denial of service attack.	✓	✓	✓	400
12.1.2	Verify that compressed files are checked for "zip bombs" - small input files that will decompress into huge files thus exhausting file storage limits.		✓	✓	409
12.1.3	Verify that a file size quota and maximum number of files per user is enforced to ensure that a single user cannot fill up the storage with too many files, or excessively large files.		✓	✓	770

12.2 V12.2 File Integrity Requirements

#	Description	L1	L2	L3	CWE

12.2.1	Verify that files obtained from untrusted sources are validated to be of expected type based on the file's content.		✓	✓	434

12.3 V12.3 File execution Requirements

#	Description	L1	L2	L3	CWE
12.3.1	Verify that user-submitted filename metadata is not used directly with system or framework file and URL API to protect against path traversal.	✓	✓	✓	22
12.3.2	Verify that user-submitted filename metadata is validated or ignored to prevent the disclosure, creation, updating or removal of local files (LFI).	✓	✓	✓	73
12.3.3	Verify that user-submitted filename metadata is validated or ignored to prevent the disclosure or execution of remote files (RFI), which may also lead to SSRF.	✓	✓	✓	98
12.3.4	Verify that the application protects against reflective file download (RFD) by validating or ignoring user-submitted filenames in a JSON, JSONP, or URL parameter, the response Content-Type header should be set to text/plain, and the Content-Disposition header should have a fixed filename.	✓	✓	✓	641
12.3.5	Verify that untrusted file metadata is not used directly with system API or libraries, to protect against OS command injection.	✓	✓	✓	78
12.3.6	Verify that the application does not include and execute functionality from untrusted sources, such as unverified content distribution networks, JavaScript libraries, node npm libraries, or server-side DLLs.		✓	✓	829

12.4 V12.4 File Storage Requirements

#	Description	L1	L2	L3	CWE
12.4.1	Verify that files obtained from untrusted sources are stored outside the web root, with limited permissions, preferably with strong validation.	✓	✓	✓	922

12.4.2	Verify that files obtained from untrusted sources are scanned by antivirus scanners to prevent upload of known malicious content.	✓	✓	✓	509

12.5 V12.5 File Download Requirements

#	Description	L1	L2	L3	CWE
12.5.1	Verify that the web tier is configured to serve only files with specific file extensions to prevent unintentional information and source code leakage. For example, backup files (e.g. .bak), temporary working files (e.g. .swp), compressed files (.zip, .tar.gz, etc) and other extensions commonly used by editors should be blocked unless required.	✓	✓	✓	552
12.5.2	Verify that direct requests to uploaded files will never be executed as HTML/JavaScript content.	✓	✓	✓	434

12.6 V12.6 SSRF Protection Requirements

#	Description	L1	L2	L3	CWE
12.6.1	Verify that the web or application server is configured with a whitelist of resources or systems to which the server can send requests or load data/files from.	✓	✓	✓	918

13. V13: API and Web Service Verification Requirements

To ensure that a verified application that uses trusted service layer APIs (commonly using JSON or XML or GraphQL) has:

- Adequate authentication, session management and authorization of all web services.
- Input validation of all parameters that transit from a lower to higher trust level.
- Effective security controls for all API types, including cloud and Serverless API

This chapter should be read in combination with all other chapters at this same level; authentication or API session management controls are not duplicated.

13.1 V13.1 Generic Web Service Security Verification Requirements

#	Description	L1	L2	L3	CWE
13.1.1	Verify that all application components use the same encodings and parsers to avoid parsing attacks that exploit different URI or file parsing behavior that could be used in SSRF and RFI attacks.	✓	✓	✓	116
13.1.2	Verify that access to administration and management functions is limited to authorized administrators.	✓	✓	✓	419
13.1.3	Verify API URLs do not expose sensitive information, such as the API key, session tokens etc.	✓	✓	✓	598
13.1.4	Verify that authorization decisions are made at both the URI, enforced by programmatic or declarative security at the controller or router, and at the resource level, enforced by model-based permissions.		✓	✓	285
13.1.5	Verify that requests containing unexpected or missing content types are rejected with appropriate headers (HTTP response status 406 Unacceptable or 415 Unsupported Media Type).		✓	✓	434

13.2 V13.2 RESTful Web Service Verification Requirements

#	Description	L1	L2	L3	CWE
13.2.1	Verify that enabled RESTful HTTP methods are a valid choice for the user or action, such as preventing normal users using DELETE or PUT on protected API or resources.	✓	✓	✓	650
13.2.2	Verify that JSON schema validation is in place and verified before accepting input.	✓	✓	✓	20
13.2.3	Verify that RESTful web services that utilize cookies are protected from Cross-Site Request Forgery via the use of at least one or more of the following: triple or double submit cookie pattern (see references), CSRF nonces, or ORIGIN request header checks.	✓	✓	✓	352
13.2.4	Verify that REST services have anti-automation controls to protect against excessive calls, especially if the API is unauthenticated.		✓	✓	779
13.2.5	Verify that REST services explicitly check the incoming Content-Type to be the expected one, such as application/xml or application/JSON.		✓	✓	436
13.2.6	Verify that the message headers and payload are trustworthy and not modified in transit. Requiring strong encryption for transport (TLS only) may be sufficient in many cases as it provides both confidentiality and integrity protection. Per-message digital signatures can provide additional assurance on top of the transport protections for high-security applications but bring with them additional complexity and risks to weigh against the benefits.		✓	✓	345

13.3 V13.3 SOAP Web Service Verification Requirements

#	Description	L1	L2	L3	CWE
13.3.1	Verify that XSD schema validation takes place to ensure a properly formed XML document, followed by validation of each input field before any processing of that data takes place.	✓	✓	✓	20
13.3.2	Verify that the message payload is signed using WS-Security to ensure reliable transport between client and service.		✓	✓	345

Note: Due to issues with XXE attacks against DTDs, DTD validation should not be used, and framework DTD evaluation disabled as per the requirements set out in V14 Configuration.

13.4 V13.4 GraphQL and other Web Service Data Layer Security Requirements

#	Description	L1	L2	L3	CWE
13.4.1	Verify that query whitelisting or a combination of depth limiting and amount limiting should be used to prevent GraphQL or data layer expression denial of service (DoS) as a result of expensive, nested queries. For more advanced scenarios, query cost analysis should be used.		✓	✓	770
13.4.2	Verify that GraphQL or other data layer authorization logic should be implemented at the business logic layer instead of the GraphQL layer.		✓	✓	285

14. V14: Configuration Verification Requirements

14.1 V14.1 Build

#	Description	L1	L2	L3	CWE
14.1.1	Verify that the application build and deployment processes are performed in a secure and repeatable way, such as CI / CD automation, automated configuration management, and automated deployment scripts.		✓	✓	
14.1.2	Verify that compiler flags are configured to enable all available buffer overflow protections and warnings, including stack randomization, data execution prevention, and to break the build if an unsafe pointer, memory, format string, integer, or string operations are found.		✓	✓	120
14.1.3	Verify that server configuration is hardened as per the recommendations of the application server and frameworks in use.		✓	✓	16
14.1.4	Verify that the application, configuration, and all dependencies can be re-deployed using automated deployment scripts, built from a documented and tested runbook in a reasonable time, or restored from backups in a timely fashion.		✓	✓	
14.1.5	Verify that authorized administrators can verify the integrity of all security-relevant configurations to detect tampering.			✓	

14.2 V14.2 Dependency

Note: At Level 1, 14.2.1 compliance relates to observations or detections of client-side and other libraries and components, rather than the more accurate build-time static code analysis or dependency analysis. These more accurate techniques could be discoverable by interview as required.

#	Description	L1	L2	L3	CWE
14.2.1	Verify that all components are up to date, preferably using a dependency checker during build or compile time. (C2)	✓	✓	✓	1026

14.2.2	Verify that all unneeded features, documentation, samples, configurations are removed, such as sample applications, platform documentation, and default or example users.	✓	✓	✓	1002
14.2.3	Verify that if application assets, such as JavaScript libraries, CSS stylesheets or web fonts, are hosted externally on a content delivery network (CDN) or external provider, Sub resource Integrity (SRI) is used to validate the integrity of the asset.	✓	✓	✓	714
14.2.4	Verify that third party components come from pre-defined, trusted and continually maintained repositories. (C2)		✓	✓	829
14.2.5	Verify that an inventory catalog is maintained of all third party libraries in use. (C2)		✓	✓	
14.2.6	Verify that the attack surface is reduced by sandboxing or encapsulating third party libraries to expose only the required behaviour into the application. (C2)		✓	✓	265

14.3 V14.3 Unintended Security Disclosure Requirements

Configurations for production should be hardened to protect against common attacks, such as debug consoles, raise the bar for cross-site scripting (XSS) and remote file inclusion (RFI) attacks, and to eliminate trivial information discovery "vulnerabilities". Many of these issues are rarely rated as a significant risk, but they are chained together with other vulnerabilities. If these issues are not present by default, it raises the bar before most attacks can succeed.

#	Description	L1	L2	L3	CWE
14.3.1	Verify that web or application server and framework error messages are configured to deliver user actionable, customized responses to eliminate any unintended security disclosures.	✓	✓	✓	209
14.3.2	Verify that web or application server and application framework debug modes are disabled in production to eliminate debug features, developer consoles, and unintended security disclosures.	✓	✓	✓	497
14.3.3	Verify that the HTTP headers or any part of the HTTP response do not expose detailed version information of system components.	✓	✓	✓	200

14.4 V14.4 HTTP Security Headers Requirements

#	Description	L1	L2	L3	CWE
14.4.1	Verify that every HTTP response contains a content type header specifying a safe character set (e.g., UTF-8, ISO 8859-1).	✓	✓	✓	173
14.4.2	Verify that all API responses contain Content-Disposition: attachment; filename="api.json" (or other appropriate filename for the content type).	✓	✓	✓	116
14.4.3	Verify that a content security policy (CSPv2) is in place that helps mitigate impact for XSS attacks like HTML, DOM, JSON, and JavaScript injection vulnerabilities.	✓	✓	✓	1021
14.4.4	Verify that all responses contain X-Content-Type-Options: nosniff.	✓	✓	✓	116
14.4.5	Verify that HTTP Strict Transport Security headers are included on all responses and for all subdomains, such as Strict-Transport-Security: max-age=15724800; includeSubdomains.	✓	✓	✓	523
14.4.6	Verify that a suitable "Referrer-Policy" header is included, such as "no-referrer" or "same-origin".	✓	✓	✓	116
14.4.7	Verify that a suitable X-Frame-Options or Content-Security-Policy: frame-ancestors header is in use for sites where content should not be embedded in a third-party site.	✓	✓	✓	346

14.5 V14.5 Validate HTTP Request Header Requirements

#	Description	L1	L2	L3	CWE
14.5.1	Verify that the application server only accepts the HTTP methods in use by the application or API, including pre-flight OPTIONS.	✓	✓	✓	749
14.5.2	Verify that the supplied Origin header is not used for authentication or access control decisions, as the Origin header can easily be changed by an attacker.	✓	✓	✓	346

14.5.3	Verify that the cross-domain resource sharing (CORS) Access-Control-Allow-Origin header uses a strict white-list of trusted domains to match against and does not support the "null" origin.	✓	✓	✓	346
14.5.4	Verify that HTTP headers added by a trusted proxy or SSO devices, such as a bearer token, are authenticated by the application.		✓	✓	306

European Maritime Safety Agency

EMSA security development recommendation table

Version 1.0
Date: 17/09/19

Document History

Version	Date	Changes	Prepared	Approved
1.0	17/09/2019	Publish	EMSA	

Table of Contents

Acronyms	
ASVS	OWASP Application Security Verification Standard

1. Secure development verification

Secure development is a requirement for any application or component that is integrated into EMSA ICT Landscape. OWASP Application Security Verification Standard is an industry standard that complies with EMSA requirements to verify that specific security measures are in place in the application or code. The aim of the following implementation table is to help with the compliance of ASVS requirements into EMSA applications by providing recommended cheat-sheets for almost every item.

1.1 V1: Architecture, design and Threat Modeling

Requirements family	Requirement	Recommended cheat sheet
V1.1 Secure Software Development Lifecycle Requirements	Threat modelling	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Threat_Modeling_Cheat_Sheet.md
	Attack surface Analysis	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Attack_Surface_Analysis_Cheat_Sheet.md
	Abuse Case	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Abuse_Case_Cheat_Sheet.md
V1.2 Authentication Architectural Requirements	Requirements come through: • OAM • OAuth2	

	• OpenID • Other such as SAML, FIDO…	

V1.3 Session Management Architectural Requirements	• Session ID properties
	• Session management implementation
	• Cookies
	• HTML5 web storage API (if applicable)
	• Session ID Life Cycle
	• Session Expiration (check EMSA Policy)
	• Additional Client-Side Defences for Session Management (if applicable)
	• Session Attacks Detection:
	  ○ Session ID guessing and brute force
	  ○ Session ID Anomalies
	  ○ Logging sessions life cycle
	  ○ Binding session ID to Other user properties
	  ○ Simultaneous session logons (if applicable)
	• Session Management WAF protections
		https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html

V1.4 Access Control Architectural Requirements	Docker Security	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Docker_Security_Cheat_Sheet.md

V1.5 Input and Output Architectural Requirements	Deserialization	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Deserialization_Cheat_Sheet.md
	Abuse Case	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Abuse_Case_Cheat_Sheet.md
V1.6 Cryptographic Architectural Requirements	Cryptographic storage	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Cryptographic_Storage_Cheat_Sheet.md
	Key management	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Key_Management_Cheat_Sheet.md
V1.7 Errors, Logging and Auditing Architectural Requirements	Logging	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Logging_Cheat_Sheet.md
V1.8 Data Protection and Privacy Architectural Requirements	Data protection: cryptography, support for HSTS, digital certificate pinning, etc.	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/User_Privacy_Protection_Cheat_Sheet.md
	Abuse Case	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Abuse_Case_Cheat_Sheet.md
V1.9 Communications Architectural Requirements	Transport layer protection	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Transport_Layer_Protection_Cheat_Sheet.md
	TLS cypher string	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/TLS_Cipher_String_Cheat_Sheet.md
V1.10 Malicious Software Architectural Requirements	N/A	

V1.11 Business Logic Architectural Requirements	Abuse case	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Abuse_Case_Cheat_Sheet.md
V1.12 Secure File Upload Architectural Requirements	Transport layer protection	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Transport_Layer_Protection_Cheat_Sheet.md
V1.13 API Architectural Requirements	• HTTPS • Access control • JWT • Restrict HTTP methods (check for Java EE) • Input validation • Validation of content types (includes XXE) • CORS • Security headers • Error handling • Audit logs • Sensitive information in HTTP requests • HTTP return code • Management endpoints	General: https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/REST_Security_Cheat_Sheet.md HTTPS: https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Transport_Layer_Protection_Cheat_Sheet.md JWT for Java: https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/JSON_Web_Token_Cheat_Sheet_for_Java.md#token-explicit-revocation-by-the-user Validation of content types: XXE attack: https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/XML_External_Entity_Prevention_Cheat_Sheet.md Security headers: https://www.owasp.org/index.php/OWASP_Secure_Headers_Project#tab=Headers Restrict HTTP methods in Java EE needs to check *bypassing web authentication and authorization with HTTP verb tampering* common misconfiguration: https://github.com/OWASP/CheatSheetSeries/blob/master/assets/REST_Security_Cheat_Sheet_Bypassing_VBAAC_with_HTTP_Verb_Tampering.pdf

V1.14 Configuration Architectural Requirements	N/A	

1.2 V2: Authentication Verification Requirements

Requirements family	Requirement	Recommended cheat sheet
V2.1 Password Security Requirement	N/A	
V2.2 General Authenticator Requirements	Authentication general approach	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Authentication_Cheat_Sheet.md
	Transport Layer Protection	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Transport_Layer_Protection_Cheat_Sheet.md
	TLS Cipher string	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/TLS_Cipher_String_Cheat_Sheet.md
V2.3 Authenticator Lifecycle Requirements	N/A	
V2.4 Credential Storage Requirements	Password storage	Password storage including implementation proposal in Java (using Argon2 library): https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Password_Storage_Cheat_Sheet.md

V2.5 Credential Recovery Requirements	Choosing and using security questions	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Choosing_and_Using_Security_Questions_Cheat_Sheet.md
	Forgot password	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Forgot_Password_Cheat_Sheet.md
V2.6 Look-up Secret Verifier Requirements	N/A	
V2.7 Out of Band Verifier Requirements	Forgot password	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Forgot_Password_Cheat_Sheet.md
V2.8 Single or Multi Factor One Time Verifier Requirements	N/A	
V2.9 Cryptographic Software and Devices Verifier Requirements	Cryptographic storage	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Cryptographic_Storage_Cheat_Sheet.md
	Key management	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Key_Management_Cheat_Sheet.md
V2.10 Service Authentication Requirements	N/A	

1.3 V3: Session Management Verification Requirements

Requirements family	Requirement	Recommended cheat sheet
V3.1 Fundamental Session Management Requirements	N/A	
V3.2 Session Binding Requirements	Session binding	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Session_Management_Cheat_Sheet.md
V3.3 Session Logout and Timeout Requirements	Session logout and timeout (check EMSA Policy for cut off sessions: 90min non-active session, 24h active sessions)	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Session_Management_Cheat_Sheet.md
V3.4 Cookie-based Session Management	Cookies	Cookies section: https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Session_Management_Cheat_Sheet.md
	Cross-Site Request Forgery Attack (CSRF attack)	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.md
V3.5 Token-based Session Management	JWT for Java	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/JSON_Web_Token_Cheat_Sheet_for_Java.md
	REST Security	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/REST_Security_Cheat_Sheet.md
V3.6 Re-authentication from a Federation or Assertion	N/A	

V3.7 Defences Against Session Management Exploit	General defences	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Session_Management_Cheat_Sheet.md
	Transaction Authorization (N/A)	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Transaction_Authorization_Cheat_Sheet.md

1.4 V4: Access Control Verification Requirements

Requirements family	Requirement	Recommended cheat sheet
V4.1 General Access Control Design	Authorization testing automation	Authorization testing automation: https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Authorization_Testing_Automation.md Access Control: https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Access_Control_Cheat_Sheet.md
V4.2 Operation Level Access Control	Insecure Direct Object Reference attack (IDOR attack) prevention	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.md
	Cross-Site Request Forgery Attack (CSRF attack)	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.md

	Authorization testing automation	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Authorization_Testing_Automation.md
V4.3 Other Access Control Considerations	Fuzzing attacks	https://www.owasp.org/index.php/OWASP_Testing_Guide_Appendix_C:_Fuzz_Vectors

1.5 V5: Validation, Sanitization and Encoding Verification Requirements

Requirements family	Requirement	Recommended cheat sheet
V5.1 Input Validation Requirements	Mass Assignment vulnerability (Improperly Controlled Modification of Dynamically-Determined Object Attributes)	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Mass_Assignment_Cheat_Sheet.md
	Input validation	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Input_Validation_Cheat_Sheet.md
	XSS attack prevention	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.md
	File upload validation	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Protect_FileUpload_Against_Malicious_File.md

V5.2 Sanitization and Sandboxing Requirements	Server Side Request Forgery attack prevention (SSRF)	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Server_Side_Request_Forgery_Prevention_Cheat_Sheet.md
	XSS attack prevention	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.md
	DOM based XSS attack prevention	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/DOM_based_XSS_Prevention_Cheat_Sheet.md
	Unvalidated Redirects and Forwards	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Unvalidated_Redirects_and_Forwards_Cheat_Sheet.md
V5.3 Output encoding and Injection Prevention Requirements	XSS attack prevention	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.md
	DOM based XSS attack prevention	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/DOM_based_XSS_Prevention_Cheat_Sheet.md

	HTML5 Security:
	• Communication APIS
	  ○ Web messaging
	  ○ Cross Origin Resource Sharing (CORS)
	  ○ Websockets implementation
	  ○ Server-sent events
	• Sotrage APIS
		https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/HTML5_Security_Cheat_Sheet.md

	○ Local storage ○ Client-side databases • Geolocation • Web workers • Tabnabbing • Sandboxed frames • Credential and Personally Identifiable Information (PII) Input • Offline applications • Progressive Enhancements and Graceful Degradation Risks • HTTP headers	
	Injection prevention	General: https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Injection_Prevention_Cheat_Sheet.md Injection prevention in Java: https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Injection_Prevention_Cheat_Sheet_in_Java.md
	Input Validation	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Input_Validation_Cheat_Sheet.md
	LDAP Injection Prevention	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/LDAP_Injection_Prevention_Cheat_Sheet.md

	OS Command Injection Defense	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/OS_Command_Injection_Defense_Cheat_Sheet.md
	Protect File Upload Against Malicious File	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Protect_FileUpload_Against_Malicious_File.md
	Query Parameterization	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Query_Parameterization_Cheat_Sheet.md
	SQL Injection Prevention	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.md
	Unvalidated Redirects and Forwards	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Unvalidated_Redirects_and_Forwards_Cheat_Sheet.md
	Bean Validation	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Bean_Validation_Cheat_Sheet.md
	XXE Prevention	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/XML_External_Entity_Prevention_Cheat_Sheet.md
	XML Security	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/XML_Security_Cheat_Sheet.md
V5.4 Memory, String, and Unmanaged Code Requirements	N/A	
V5.5 Deserialization Prevention Requirements	Deserialization	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Deserialization_Cheat_Sheet.md

	XXE Prevention	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/XML_External_Entity_Prevention_Cheat_Sheet.md
	XML Security	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/XML_Security_Cheat_Sheet.md

1.6 V6: Stored Cryptography Verification Requirements

Requirements family	Requirement	Recommended cheat sheet
V6.1 Data Classification	Abuse Case	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Abuse_Case_Cheat_Sheet.md
	User Privacy Protection	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/User_Privacy_Protection_Cheat_Sheet.md
V6.2 Algorithms	Cryptographic Storage	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Cryptographic_Storage_Cheat_Sheet.md
	Key Management	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Key_Management_Cheat_Sheet.md
V6.3 Random Values	N/A	
V6.4 Secret Management	Key Management	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Key_Management_Cheat_Sheet.md

1.7 V7: Error Handling and Logging Verification Requirements

Requirements family	Requirement	Recommended cheat sheet
V7.1 Log Content Requirements	Logging	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Logging_Cheat_Sheet.md
V7.2 Log Processing Requirements	Logging	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Logging_Cheat_Sheet.md
V7.3 Log Protection Requirements	Logging	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Logging_Cheat_Sheet.md
V7.4 Error Handling	Error Handling	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Error_Handling_Cheat_Sheet.md

1.8 V8: Data Protection Verification Requirements

Requirements family	Requirement	Recommended cheat sheet
V8.1 General Data Protection	Data Protection Impact Assessment	Ask EMSA for DPIA template
V8.2 Client-side Data Protection	N/A	
V8.3 Sensitive Private Data	Data Protection Impact Assessment	Ask EMSA for DPIA template

1.9 V9: Communications Verification Requirements

Requirements family	Requirement	Recommended cheat sheet
V9.1 Communications Security Requirements	HTTP Strict Transport Security	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/HTTP_Strict_Transport_Security_Cheat_Sheet.md
	Transport Layer Protection	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Transport_Layer_Protection_Cheat_Sheet.md
	TLS Cipher String	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/TLS_Cipher_String_Cheat_Sheet.md
V9.2 Server Communications Security Requirements	N/A	

1.10 V10: Malicious Code Verification Requirements

Requirements family	Requirement	Recommended cheat sheet
V10.1 Code Integrity Controls	Third Party Javascript Management	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Third_Party_Javascript_Management_Cheat_Sheet.md
V10.2 Malicious Code Search	N/A	
V10.3 Deployed Application Integrity Controls	Docker Security	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Docker_Security_Cheat_Sheet.md

1.11 V11: Business Logic Verification Requirements

Requirements family	Requirement	Recommended cheat sheet
V11.1 Business Logic Security Requirements	Abuse Case	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Abuse_Case_Cheat_Sheet.md

1.12 V12: File and Resources Verification Requirements

Requirements family	Requirement	Recommended cheat sheet
V12.1 File Upload Requirements	Protect File Upload Against Malicious File	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Protect_FileUpload_Against_Malicious_File.md
V12.2 File Integrity Requirements	Protect File Upload Against Malicious File	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Protect_FileUpload_Against_Malicious_File.md

	Third Party Javascript Management	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Third_Party_Javascript_Management_Cheat_Sheet.md
V12.3 File execution Requirements	N/A	
V12.4 File Storage Requirements	N/A	
V12.5 File Download Requirements	N/A	
V12.5 File Download Requirements	Server Side Request Forgery (SSRF) attack Prevention	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Server_Side_Request_Forgery_Prevention_Cheat_Sheet.md
	Unvalidated Redirects and Forwards	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Unvalidated_Redirects_and_Forwards_Cheat_Sheet.md

1.13 V13: API and Web Service Verification Requirements

Requirements family	Requirement	Recommended cheat sheet
V13.1 Generic Web Service Security Verification Requirements	Web Service Security	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Web_Service_Security_Cheat_Sheet.md
	Server Side Request Forgery (SSRF) attack Prevention	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Server_Side_Request_Forgery_Prevention_Cheat_Sheet.md

V13.2 RESTful Web Service Verification Requirements	REST Assessment	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/REST_Assessment_Cheat_Sheet.md
	REST Security	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/REST_Security_Cheat_Sheet.md
	Cross-Site Request Forgery (CSRF) attack Prevention	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.md
V13.3 SOAP Web Service Verification Requirements	XML Security	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/XML_Security_Cheat_Sheet.md
V13.4 GraphQL and other Web Service Data Layer Security Requirements	N/A	

1.14 V14: Configuration Verification Requirements

Requirements family	Requirement	Recommended cheat sheet
V14.1 Build	Docker Security	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Docker_Security_Cheat_Sheet.md
V14.2 Dependency	Vulnerable Dependency Management	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Vulnerable_Dependency_Management_Cheat_Sheet.md
	Docker Security	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Docker_Security_Cheat_Sheet.md

V14.3 Unintended Security Disclosure Requirements	Error Handling	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Error_Handling_Cheat_Sheet.md
V14.4 HTTP Security Headers Requirements	Content Security Policy	https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Content_Security_Policy_Cheat_Sheet.md
V14.5 Validate HTTP Request Header Requirements	N/A	

European Maritime Safety Agency

JASPER Horizontal Platform

Technical Overview

EMSA

Document History

Title	**Jasper Horizontal Platform. Technical Overview**
Version	1.3 from 06/11/2019

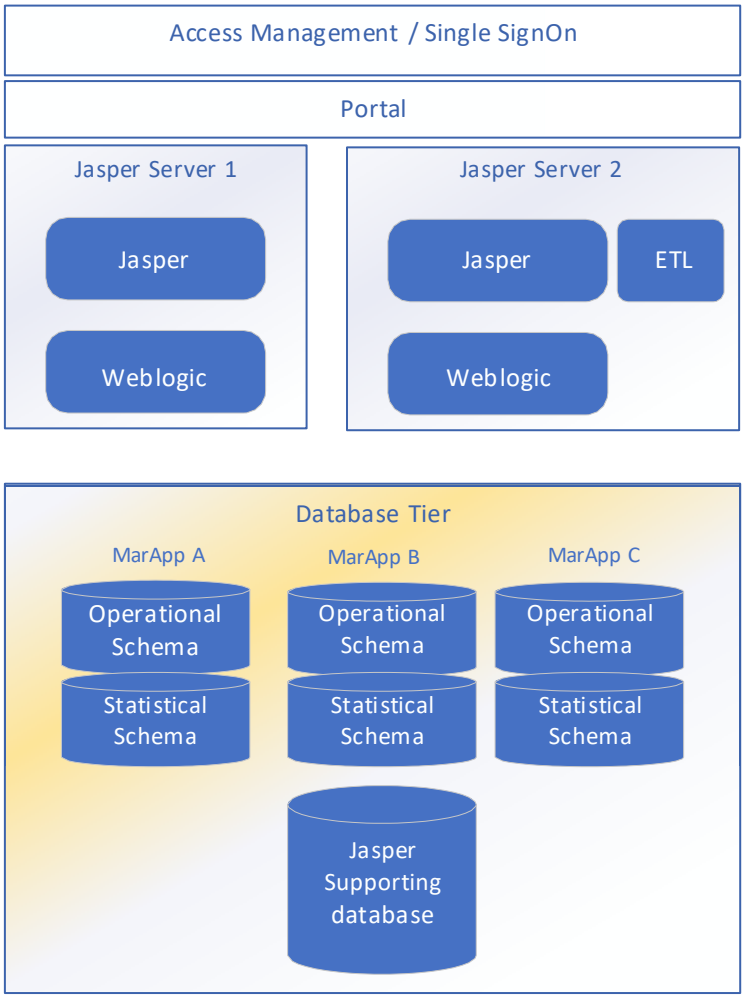
TABLE OF CONTENTS

1. Introduction

This document provides an overview of EMSA's Jasper Horizontal Platform and indications of the steps needed to add new reporting modules to this Horizontal Platform.

2. Overview of EMSA's Jasper Horizontal Platform

EMSA's Jasper Horizontal Platform is a JasperSoft installation configured and tuned to work closely integrated in EMSA's environments. It serves as a Business Intelligence and Reporting platform providing these services to EMSA's Maritime Applications (MarApps).
The next figure depicts Jasper Horizontal Platform:



- Two servers running Linux Redhat V7.4 are the base infrastructure for EMSA's Jasper Horizontal Platform. Those servers are identified in the figure above as "Jasper Server 1" and "Jasper Server 2";
- JasperSoft V7.1 is deployed in a cluster of 2 Weblogic 12c application servers and having the supporting database running in the database tier (see below);
- The database tier is implemented on top of Oracle DBMS 12c;
- Each MarApp that uses Jasper Services will have at least 2 different schemas: the Operational schema used by the MarApp and the Statistical Schema used by Jasper;
- ETL processes are responsible to Extract information from the Operational schema, Transform and Load into the Statistical Schema;

- It should be noted that we currently have several types of ETL processes, from the simplest ones using view, materialized views or PL/SQL procedures to more complex ETL processes running on top of Talend V6.3.1 (part of the JasperSoft suite);
- User accesses to Jasper functionalities are controlled/managed through the EMSA's Access Management and Single Sign-On;
- Jasper Web interfaces are usually available in EMSA's Portal.

Please note that versions indicated are the ones currently deployed in EMSA environments. However, due to EMSA's patching policies, they might change over time. Final versions to be considered in any future development shall be agreed at the Project Kick-off meeting.

3. Integration of new Reporting Modules

New reporting modules can be added to Jasper Horizontal platform following the integration process described in next chapters.
It must be noted that the steps described here are a guideline for standard Reporting Modules. However, new needs can be discussed but they have to be fully detailed and justified. EMSA will assess the need and the new request being made and will take a decision on how to proceed (accepting or rejecting).

3.1 LDAP configuration

Any standard new application that will be available in Jasper will have two Roles:

- ROLE_APPLICATIONNAME_RW;
- ROLE_APPLICATIONNAME_RO;

In LDAP these two roles are represented by two groups, their members will have the permissions that the role has in Jasper.

To add a new report module or application it is necessary to create two new groups and add the users to those groups. The name of the groups will be:

- JASPER_APPLICATIONNAME_RW;
- JASPER_APPLICATIONNAME_RO;

3.2 Jasper Server

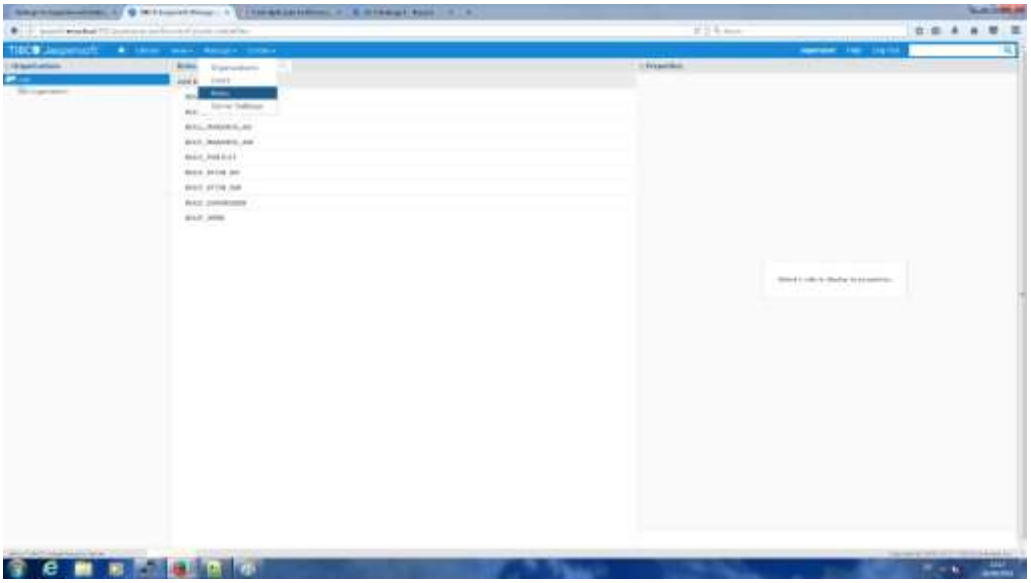
This chapter will show the necessary steps to configure a new application in Jasper Server.

3.2.1 Roles

It is necessary to have two Roles in Jasper Server. These roles represent the two groups created before in LDAP. The following steps are necessary to create the necessary roles:

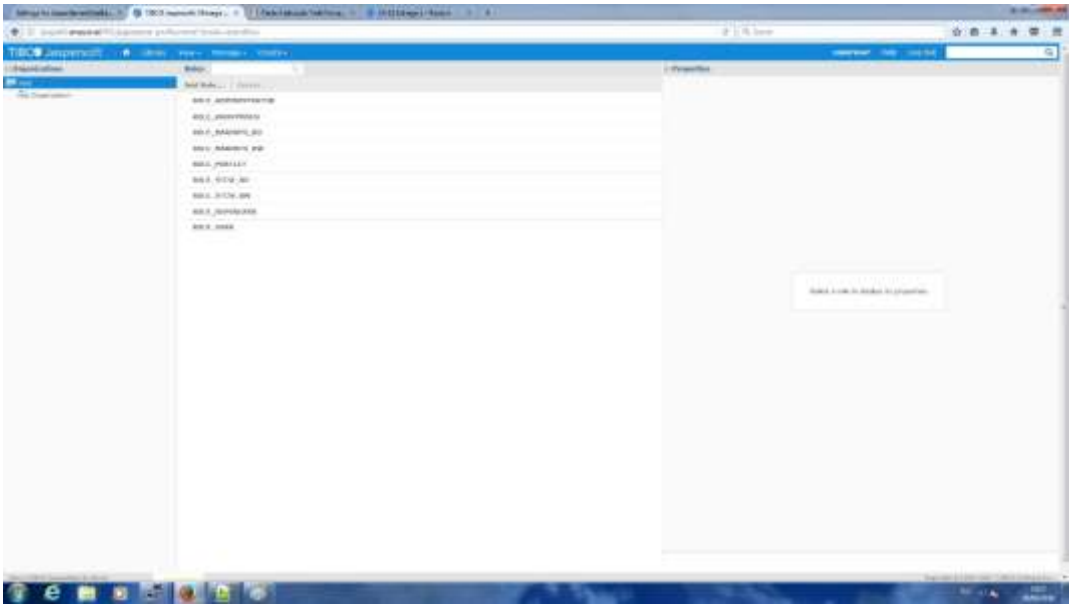
3.2.1.1 Access Roles Menu

After login, click in menu Manage -> Roles



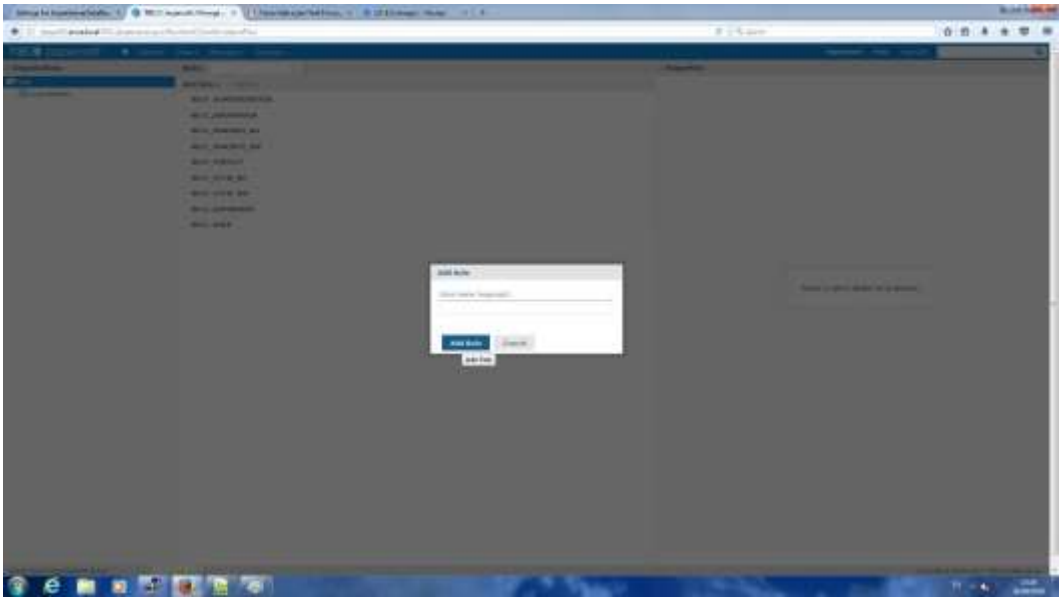
3.2.1.2 Add Roles

Click in Add Role button to create new Roles.



And add the two new Roles to be created:

- ROLE_APPLICATIONNAME_RW;
- ROLE_APPLICATIONNAME_RO;



3.2.2 Repository

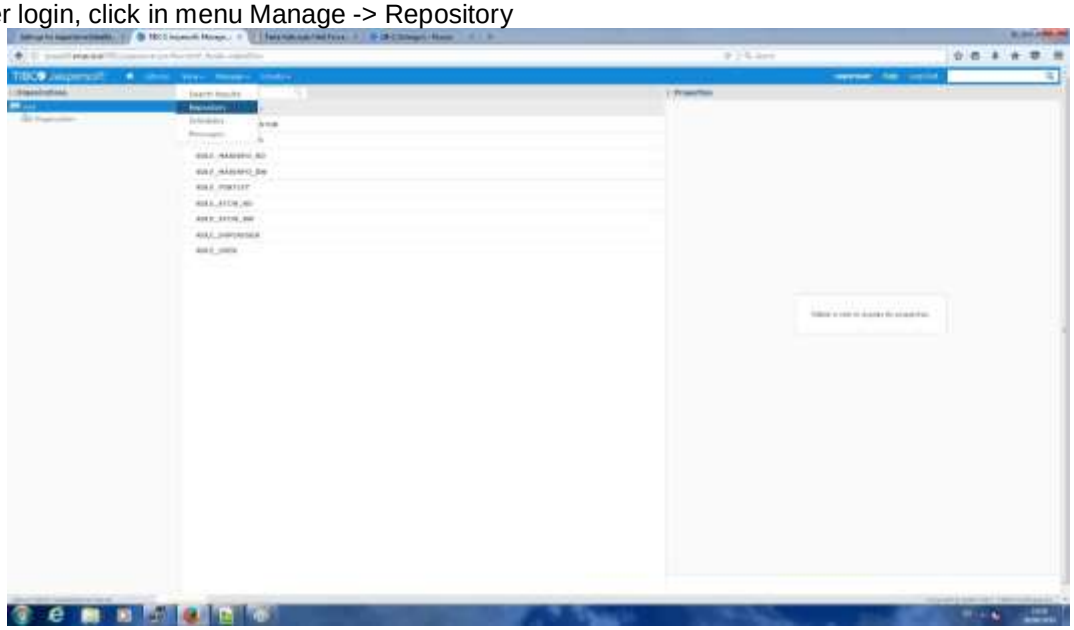
In the repository three folders will exist:

- DATA SOURCES;
  - This folder will have the necessary data sources to the required domains, each data source needs to be created in WebLogic application server and then created here with JNDI reference;
- DOMAINS;
  - This folder will have the necessary domains to the required reports or adhoc views;
- RESTRICTED;
  - This folder will have the required reports or adhoc views;

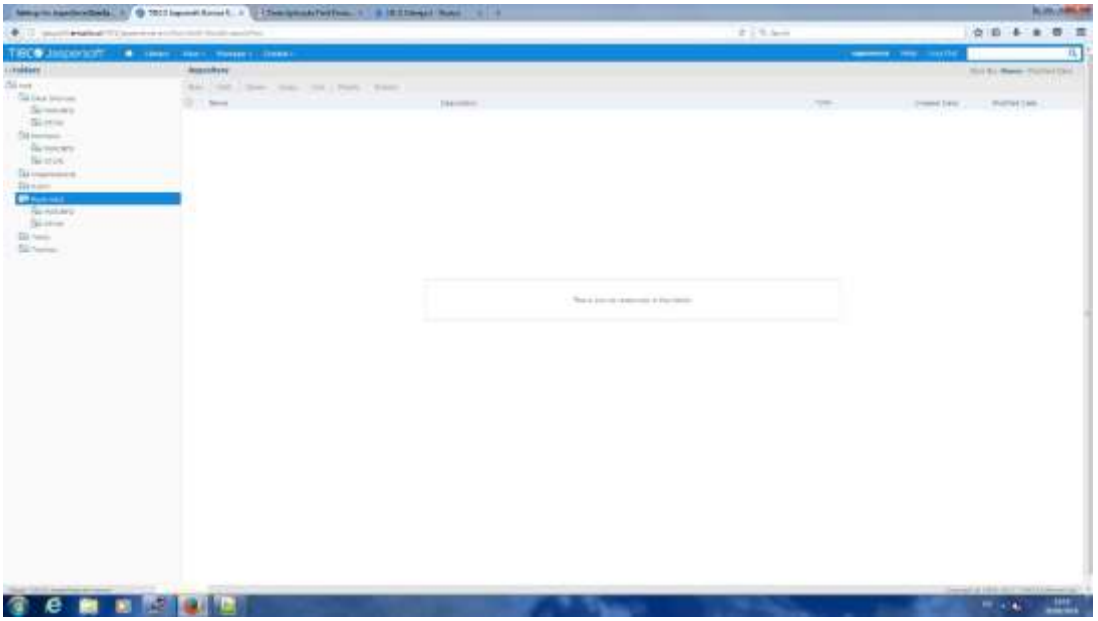
Each of these folders will have one folder named APPLICATION_NAME that will contain the respective resources for the application.

To access the repository, reproduce the followed steps:

1. After login, click in menu Manage -> Repository



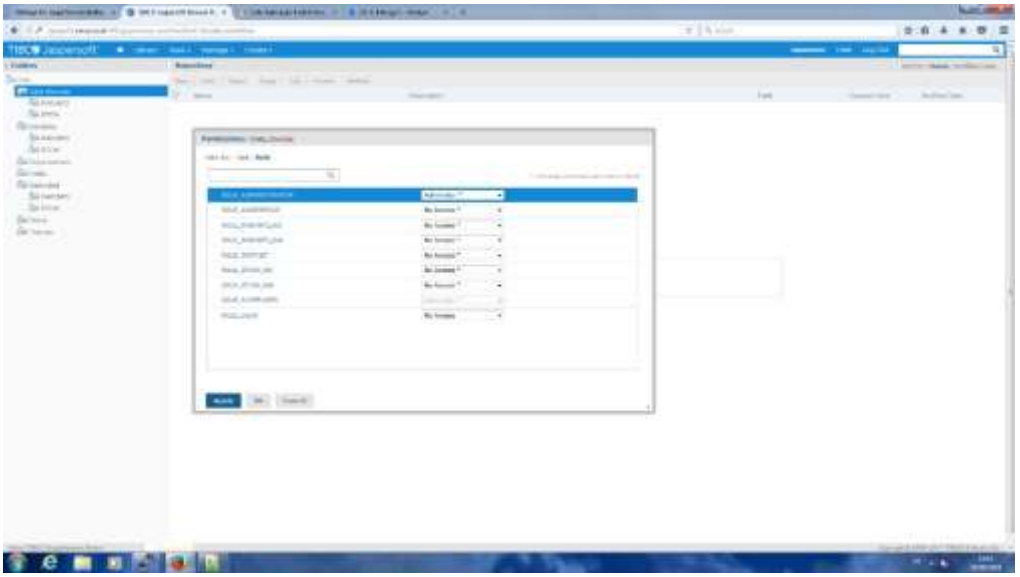
2. The folder structure reference before will stay like the followed picture



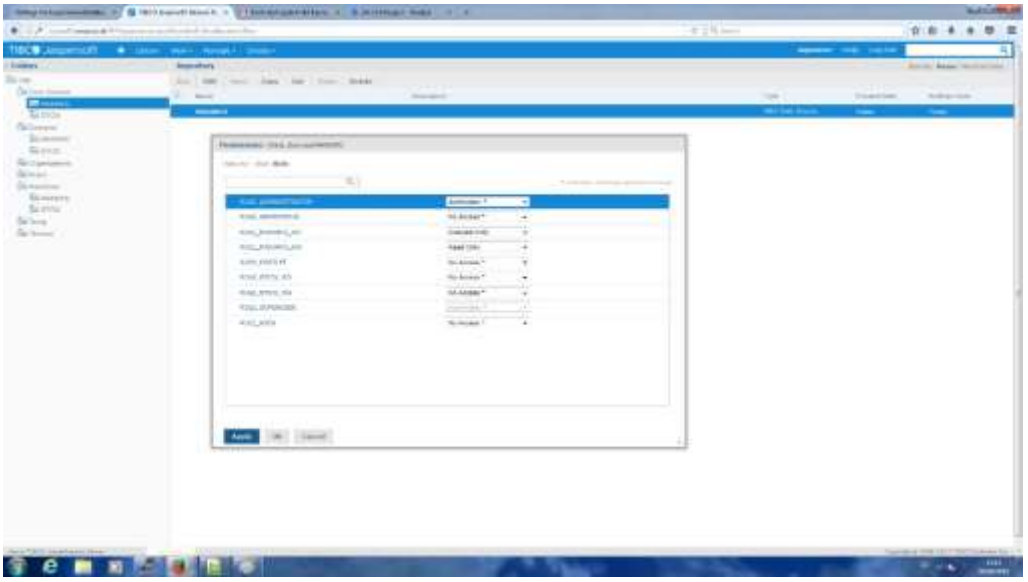
3.2.2.1 Data Sources

Under the folder Data Source it is necessary to create a folder to the required Application.

The following images present the permissions under the folder Data Sources.



The folder data source will have **No Access** permission to ROLE_APPLICATIONNAME_RW and ROLE_APPLICATIONNAME_RO.

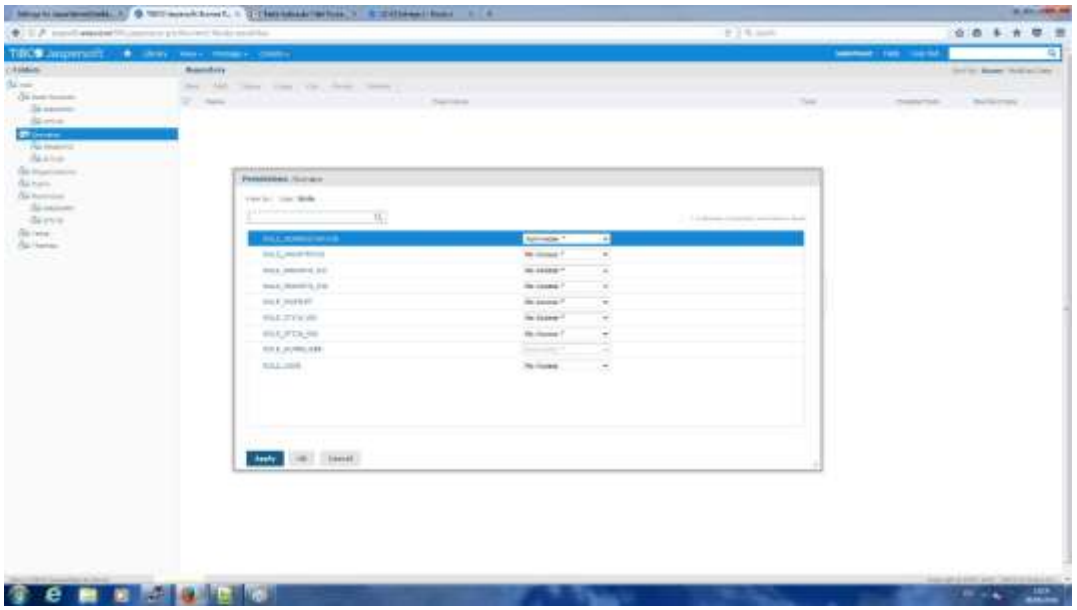


The folder data source -> APPNAME will have **Execute Only** permission to ROLE_APPLICATIONNAME_RO and **Read Only** permission to ROLE_APPLICATIONNAME_RW.

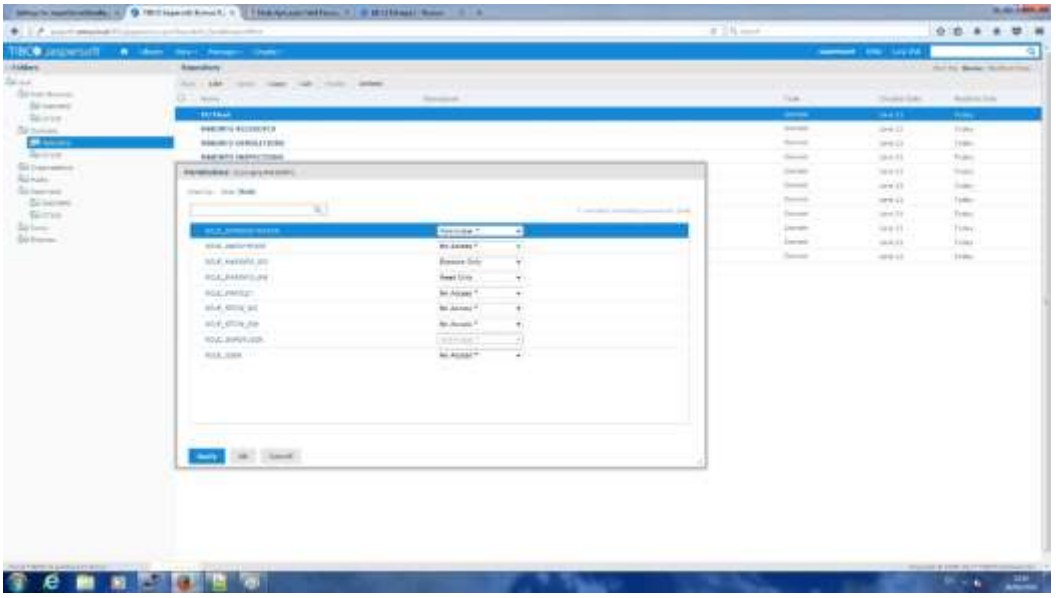
3.2.2.2 Domains

Under the folder Domains it is necessary to create a folder to the required Application.

The following images present the permissions under the folder Domains.



The folder domains will have **No Access** permission to ROLE_APPLICATIONNAME_RW and ROLE_APPLICATIONNAME_RO.

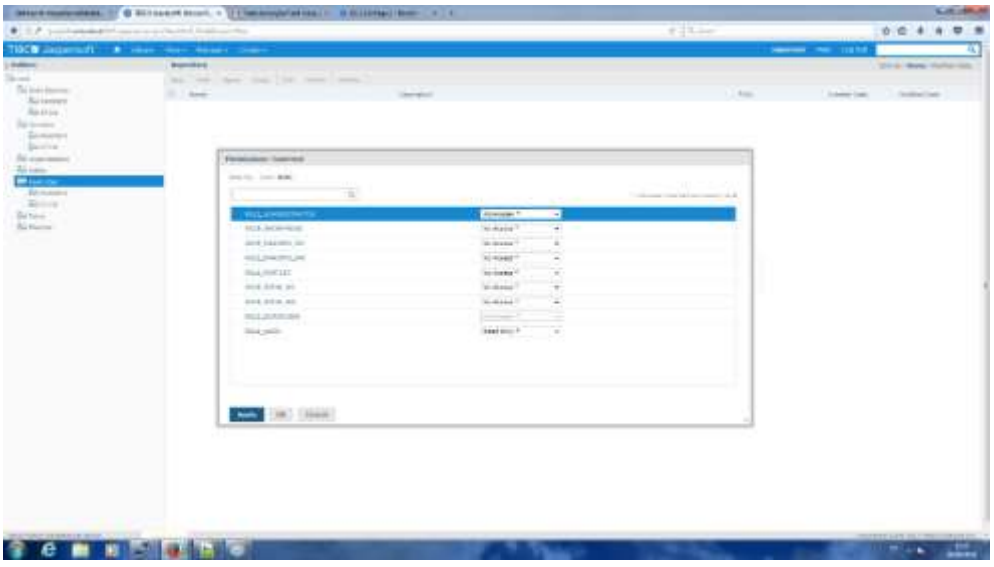


The folder domains -> APPNAME will have **Execute Only** permission to ROLE_APPLICATIONNAME_RO and **Read Only** permission to ROLE_APPLICATIONNAME_RW.

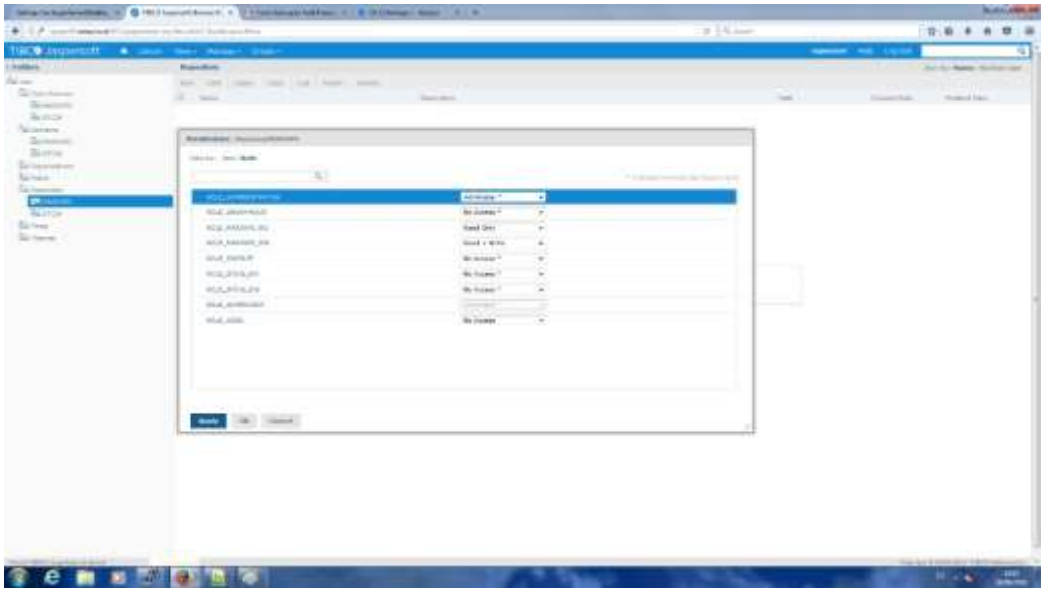
3.2.2.3 Restricted

Under the folder Restricted it is necessary to create a folder to the required Application.

The following images present the permissions under the folder Restricted.



The folder restricted will have **No Access** permission to ROLE_APPLICATIONNAME_RW and ROLE_APPLICATIONNAME_RO.



The folder restricted -> APPNAME will have **Read Only** permission to ROLE_APLICATIONNAME_RO and **Read+Write** permission to ROLE_APPLICATIONNAME_RW.

4. Delivery Package

Any new report module must be delivered to EMSA in a “Delivery Package” containing all documentation and resources needed to deploy it at EMSA environments. The “Delivery Package” shall contain as a minimum:

- Roles definition and permissions
- Datasource definitions and creation scripts
- Database package
  - Full set of script (creates schemas from scratch)
  - Version scripts (updates schemas from version n to version n+1)
- Jasper Domains
- Adhoc views, Reports and Dashboards

[End of Document]

ABOUT THE EUROPEAN MARITIME SAFETY AGENCY

The European Maritime Safety Agency is one of the European Union's decentralised agencies. Based in Lisbon, the Agency provides technical assistance and support to the European Commission and Member States in the development and implementation of EU legislation on maritime safety, pollution by ships and maritime security. It has also been given operational tasks in the field of oil pollution response, vessel monitoring and in long-range identification and tracking of vessels.

European Maritime Safety Agency
Praça Europa 4
1249-206 Lisbon, Portugal
Tel +351 21 1209 200
Fax +351 21 1209 210
emsa.europa.eu

EMSA

European Maritime Safety Agency

EMSA SOA Guidelines & Rules

Draft - 20190926

Date:02/12/2019

EMSA

Document History

Version	Date	Changes	Prepared	Approved
Final	02/12/2019	Publish	EMSA	

Table of Contents

List of Abbreviations

ATLAS	EMSA Architecture Repository
SOAP	Simple Object Access Protocol
REST	Representational State Transfer
JMS	Java Messaging Service
HTTP	Hypertext Transport Protocol
XML	eXtensible Mark-up Language
JSON	JavaScript Object Notation
API	Application Programming Interface
APCG	ICT Architecture and Planning Coordination Group
RPC	Remote Procedure Call
RMI	Remote Method Invocation
IIOP	Internet Inter-Orb Protocol
COM	Component Object Model
TPM	Technical Project Manager
ESB	Enterprise Service Bus
CAB	Change Authority Board
APCG	Architecture and Planning Coordination Group
ICT-SG	ICT Steering Group

1. Introduction

This document contains a set of rules that should be followed by Project Managers during the design of system interfaces in order to allow for a better integration within the SSN Ecosystem. EMSA Maritime Applications will need to comply with the rules defined below, whilst for corporate applications we recommend these rules to be applied.

2. Rules

2.1 Service Naming Convention

Use the common naming format to identify your services within the SSN Ecosystem.

Each service is identified by a “Service ID”. A Service ID indicates the “function” of the Service within the SSN Ecosystem, by means of a well-known acronym or abbreviation of the underlying information or message content. For the sake of example, the ID of the service that provides information on Countries shall start by the acronym of the application or solution, the Central Country Database, that implements the service followed by ‘/’ (i.e. CCD/) followed by the name of the service The name of the service shall be indicative of the information or function provided by the service.

A Service ID is unique, i.e. a Service ID should identify one and only one Service.

A Service ID is a string with the following format:

Maximum Length	50 characters
Character set	[A-Za-z0-9\-\._]
Examples	CCD/CountryInfo EO/IMAGE

2.2 Service Versioning

Services should be implemented using versioning to enable:

- Evolution of services without concerns for dependencies
- Stability of dependent services

This approach should be applied to all new services or updates. To allow continuity and at the same time ensuring stability of dependent services and systems, at least 1 year of backwards compatibility, starting at the date on which a new, non-compatible version is released in production, needs to be maintained. This 1-year stability period can only be reduced in case ALL service consumers agree. In order to assure the timely adaptation of clients, the TPM shall inform the TPMs of all registered – in Atlas – clients as soon as the decision to create a new version of a service, the TPM shall indicate if he expects this new version will or will not be backwards compatible[1]. He shall also distribute the ICD for the new service version as soon it has been approved

The versioning schema to be used is defined in ‘Appendix B – EMSA guidelines on use of web services and information exchange’. As described in the appendix each service shall have at least a major and a minor version number. Optionally it can also have a third part indicating build number or bug fix version.

[1] See e.g. https://www.gcloud.belgium.be/rest/#api-evolution for guidelines on how to evolve a REST + JOSN service without breaking backwards compatibility

A service's major version shall only change if there is a modification to the interface, i.e. in case the back-end implementation changes but the interface remains the same, the version number will not change. In such case the TPM shall ensure that the behaviour of the service does not change in a way that requires changes to its clients.

In any services that are accessible over a URL, the major version number should always be part of the URL. None of the other parts of the version number may be part of the URL as that may require existing clients to update whenever a minor change to the version occurs.

2.3 Data Format

For data exchange between applications / services, the data formats defined in 'Appendix A: Data Formats for data exchange between EMSA Maritime Applications' shall be the preferred choice.

In case an EMSA defined data format (CDF)[2] [3] needs adaptation to be able to support the use case for a new service, the TPM shall request a review to APCG. Where no EMSA defined Canonical Format has been defined, the use of industry accepted standards, such as those defined by OGC, OASIS, W3C, etc. shall be preferred.

2.4 Central Reference Databases

Countries, Organizations, Ports, and Geographical Areas: use of the common databases (CSD, CCD, COD, CLD, and CGD) for reference purposes is mandatory when dealing with this type of data.

Projects need to agree with the reference database implementations whether they will use the subscription service – thus listening for changes announced by the reference DB and updating the internal data accordingly – or if they will directly query the reference data using the available service interfaces. This choice will involve a trade-off between effort, performance and availability. The TPM of the reference DB application shall be involved before deciding the integration style.

2.5 Service and Application Monitoring

To harmonize and improve the monitoring of System Incidents and SLA compliance reporting, all applications shall have a single endpoint available that reports the status of the application in an unambiguous way. If the application is made up of multiple components (typically at least there is an application server and a database), the status of each of these components should be represented, either as part of the return message or by providing a separate endpoint per component. For each of the components of the application the following data shall be returned:

- Name of the component
- Version of the component
- Status, specified by one of the following values: OK, WARNING, FAILED

The standard format for application monitoring information is specified in 'Appendix C: Application Monitoring Format'.

If an application is deployed on a webserver/application server, it shall have a http endpoint that can be used for monitoring purposes. The health checks for all individual components should be combined on a single html page and this page should avoid the use of any client-side scripting and not rely on any graphical representation of the status.

If the application is not deployed on a web server another method for checking the health needs to be provided; this could be for example a JMX server or a Unix script allowing to reliably check the health of each service/component shall be delivered.

[2] For XML CDF formats see the latest tagged release in http://pforge01.emsa.local/svn/repos/cdf/tags/
[3] For AVRO CDF formats see master branch of https://gitlab.com/emsa/cdf/avro

2.6 System-to-System Interfaces

The guidelines given in 'Appendix B: EMSA guidelines on use of web services and information exchange' shall be followed for all system to system service interfaces.

The SLA of the service should be specified such that it can be monitored via ESB operational monitoring over a given time window.

Service Type	Aggr. Interval	Avg. Resp. Time	Messages	Errors	SLA Alerts	Pipeline Alerts	Endpoint URI Status
Proxy Service	0 hr(s) 10 mins	0 msecs	0	0	0	0	N/A
Proxy Service	0 hr(s) 10 mins	0 msecs	0	0	0	0	N/A
Proxy Service	0 hr(s) 10 mins	0 msecs	0	0	0	0	N/A
Proxy Service	0 hr(s) 10 mins	0 msecs	0	0	0	0	N/A
Business Service	0 hr(s) 10 mins	0 msecs	0	0	0	N/A	Online
Proxy Service	0 hr(s) 10 mins	0 msecs	0	0	0	0	N/A

Figure 1 Available Service Health metrics in OSB

Operational monitoring metrics for the service shall be activated and monitored via NAGIOS, allowing to report on service SLA compliance.

The specification for the service shall include:

- Aggregation interval for the service. Unless there is a specific reason for deviation, e.g. a very high or very low frequency of calls to the service, this should be 10 minutes.
- Average response time over the aggregation interval
- Messages: maximal and minimal number of invocations of the service over the aggregation interval
- Errors: Acceptable number of errors over the aggregation interval

The BRAT shall include a section detailing the changed/new interfaces resulting from the requested change. No direct connections between applications shall be allowed without prior consultation of APCG. Whenever a new interface between applications is created or an existing interface is modified, the service shall be proxied through ESB.

Each interface shall be put in the EMSA Architecture Repository, including information on the client applications, SLA, main technical characteristics. The TPM of the application implementing the interface is responsible for registering the service in ATLAS and keeping its information up-to-date. The TPMs of each application using the interface are responsible for registering their use of the interface in ATLAS.

The Project Plan should have a milestone where the service interface is documented (ICD). The ICD is reviewed and approved by the client application TPMs, in cooperation with the client application development contractor. Only TPMs that have registered their application(s) as a client, will be notified and consulted when there are planned changes to the interface. A reasonable time limit for reviewing the changes will be set, in order not to delay too much the design and implementation of applications. If no agreement can be reached within this period, the issue will be raised to APCG by the TPM of the application implementing the service.

2.7 Authentication and authorisation

System-to-System interfaces shall always be subject to Authentication and Authorization.

An accountID is authorized to use a System-to-System interface if it is authenticated and authorized to do so.

2.7.1 System Accounts

For accessing System-to-System interfaces, applications shall use the concept of a "System Account" defined in IdM-V2. "System Accounts" are stored on a dedicated LDAP branch and are not mixed with "Human Accounts".

Authentication shall succeed for an active "System Account" and a matching password.

2.7.2 System Roles

“System Roles” are also segregated; they are not mixed with “Human Roles”.

Each application shall define a set of “System Roles” that can only be associated to “System Accounts” (in IdM-V2, through Profiles)

Authorization shall succeed if the accountID is member of the role (or roles) that grants permission to use the System-to-System interface

2.8 Testing

Client applications define the expected behaviour through test cases (“Client Test Cases”) which are implemented (e.g. by test contractor) and delivered to the service development contractor in due time. The test report shall include the results of the execution of Client Test Cases.

The application contractor developing a service shall, at the earliest possible time after agreement of the ICD, deliver a mock implementation of the service that can be used by client applications to prepare the integration.

Appendix A Data Formats for data exchange between EMSA Maritime Applications

Message	Use case	Recommended format	External link
Vessel Positions	• Low frequency of messages • Human readable • Exchange with 3rd parties, especially those with lower technical proficiency	Position CDF (XML)	
Vessel Positions	• High frequency of messages	Position CDF (AVRO)	
Static and Voyage information	• Low frequency of messages • Human readable • Exchange with 3rd parties, especially those with lower technical proficiency	VoyageInfo CDF (XML)	
Static and Voyage information	• High frequency of messages	VoyageInfo CDF (AVRO)	
Vessel Particulars	• Low frequency of messages • Human readable • Exchange with 3rd parties, especially those with lower technical proficiency	ShipParticulars CDF (XML)	
Geographical Data	• Display in Google Earth	KML	
Geographical Data	• Data exchange between services	GML	
Alerts	• Exchange of alert information between applications	Common Alerting Protocol (CAP, XML)	

Appendix B EMSA guidelines on use of web services and information exchange

As EMSA's maritime applications are and will be composed of services implemented by many different teams and contractors, that today do not always collaborate optimally, there is a need to harmonise the architectural styles within these applications. One of the aspects of architectural style is how services/applications exchange data internally or with external system. Architecture Planning and Coordination Group (APCG) chose to address this subject first as the interchange of information has the greatest and most immediate impact on all applications composing the SSN ecosystem.

This appendix tries to give a brief overview of the potential ways to exchange data between systems, presenting pros and cons of each and provide a guideline for when to use what integration style.

1. Methods of exchanging data between applications, Integration Styles.

1.1 Simple Object Access Protocol (SOAP) web services

Description

SOAP is a protocol, is strongly typed and has a strict specification. SOAP is not limited to HTTP (e.g. JMS or SMTP may be used a transport mechanism). End-to-end security is supported through WS-* specifications, in contrast to REST where federated security (e.g. ADFS) is still work in progress (e.g. OpenID). SOAP has support for distributed, two-phase commit transactions, using WS-Atomic Transactions.

There are many different WS-* specifications, multiple WS-* can address the same issue, so it is not always clear when to use which one and a particular WS-* spec could not be supported by all vendors. In limited cases, small differences in implementation between infrastructure vendors can cause interoperability problems.

SOAP uses interfaces and named operations to expose business logic as opposed to REST which exposes resources.

SOAP has a set of standard specifications. WS-Security is the specification for security in the implementation. It is a detailed standard providing rules for security in application implementation. Like this we have separate specifications for messaging, transactions, etc. Unlike SOAP, REST does not have dedicated concepts for each of these.

Applicability

Recommended integration style for API style integration when there is need for a tight control over the interface. This is most applicable when integrating with parties outside of EMSA or when contract negotiation will be applicable on changes to the interface.

SOAP will also be the preferred integration style if additional functionality such as sending replies to different endpoints, asynchronous invocations, transactionality or other WS-* features are required.

If combined with using JMS for message exchange it can provide further decoupling of applications and increased reliability and scalability.

1.2 REST web services (XML payload)

Description

REST stands for Representational State Transfer and is not a protocol but rather an architectural style. Due to this, more attention needs to be paid to the quality of the implementation as opposed to web services using SOAP. Specifically the API design (REST URIs need to be resources, not methods), proper usage of HTTP verbs (GET to

query resources, POST to create a new resource, PUT to update a resource and DELETE to delete a resource), discoverability (e.g. messages should contain links to next action(s) to be performed) and re-use of standard HTTP/web technologies (e.g. use HTTP authentication instead of implementing an own custom authentication protocol for the service, allow for caching of GET requests, …). For web services providing a business method (e.g. performing a calculation based on inputs) SOAP will usually be a better match. REST on the other hand may be a more natural fit for exposing information (resources).

For all REST services implemented by EMSA maritime applications, an OpenAPI 3.0[4] (or later) specification will be needed.

- Whole of the web works based on REST style architecture. Consider a shared resource repository and consumers access the resources.

- REST messages should be self-contained and should help consumer in controlling the interaction between provider and consumer (example, links in message to decide the next course of action). But SOAP doesn't has any such requirements.

- REST does not enforce message format as XML or JSON or etc.

REST follows stateless model. SOAP has specifications for stateful implementation as well.

Applicability

Recommended integration style for API style integration when EMSA has a firm control over the interface and changes can be agreed on a less formal level (i.e. a simple agreement by the CAB will be enough).

Using XML as the message format is recommended rather than using JSON format as there is better tool support for validation and transformation of messages. No industry standards exist for documenting or specifying the content of a JSON message.

All services providing a REST interface must create and maintain a OpenAPI document describing all supported operations.

1.3 REST Web services (JSON payload)

Description

All recommendations and cautions mentioned for REST Web Services (XML) apply here as well. Additionally, JSON format has more limited tool support, e.g. no validation of the message against DTD or Schema, no standards for message transformation such as to XSLT or XQuery.

Applicability

Allowed integration style for API style integration when EMSA has a firm control over the interface and changes can be agreed on a less formal level. The recommendation is to use this only for integration between components of the same application.

Can be used instead of the XML message payload, if either:

1. interface will be consumed in a web browser or
2. message size is *very* important, but you cannot use AVRO or protocol buffers

[4] https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.0.md

1.4 Java Messaging Services (JMS)

Description

JMS can be used to achieve a loose coupling between applications exchanging data, allows for good scalability and reliable message delivery by relying on the underlying messaging infrastructure.

JMS based integrations will typically be more suitable for asynchronous communications.

The drawbacks are the additional complexity in setting up and configuring the messaging infrastructure.

Applicability

Recommended integration style for event driven data exchange between applications deployed in a Java Application Server.

1.5 Shared database tables

Description

Uses tables in schema accessed by multiple applications for exchanging data.

Applicability

Creates a tight coupling between applications.

This integration style is not acceptable at EMSA, except for temporary integration between 2 application and only after consultation of APCG. The reasons for choosing this integration style should always be presented to the APCG which will then provide a recommendation to the ICT Steering Group.

1.6 Remote procedure calls (RPC), CORBA, COM …

Description

These integration methods provide integration based on an API to be called by the client application. Whilst these may have performance benefits, especially if both applications run within the same machine or VM, it also reduces interoperability, e.g. both applications will usually need to be implemented in the same technology (Java IIOP, COM) or use proprietary infrastructure (CORBA).

Applicability

In exceptional cases this may be an acceptable method of sharing information; however, the reasons for choosing this integration method should be presented to the APCG which will then provide a recommendation to the ICT Steering Group.

2. Message formats

Generally, there are 4 options when defining the message format for exchange of data between applications: XML, JSON, CSV or Binary objects.

2.1 Human readable formats

2.1.1 eXtensible Mark-up Language (XML)

Description

Using XML for exchanging data has the following main benefits:

- Human readable (useful for debugging, problem resolution).
- Supported by all major programming language and platforms.
- Commonly available developer skill.
- Message format is well defined. Strong support for message specification through either XSD (preferred) or DTD. This allows both validation (at run time) and documentation using a standard syntax.
- Cross platform / technology independent, e.g. 1 service implemented in PHP and running on a Windows Server can exchange messages with a Java application hosted on a Unix server.
- Good tool support: schema editors, validator, transformation, etc.

Disadvantages:

- XML has as its major disadvantage that it can be very verbose and therefore may be less suitable for exchanging short messages at a very high rate.

When defining an XML format, the best practice at EMSA should be to be as explicit as possible. This will enhance the understanding of the interface and reduce implementation errors and complexity. On the other hand, being explicit may require interface version to be updated more frequently. Yet, such updates will likely require changes or at least testing for compatibility with all of the service's client, so forcing an explicit version upgrade should be considered an advantage.

Applicability

Should be the preferred choice when exchanging large datasets at a low rate.

Is also the recommended format when exchanging data with 3rd parties as the possibilities for validation of the data and format are more advanced and more well-known than for the other formats described in this document.

2.1.2 JavaScript Object Notation (JSON)

Description

Using JSON for exchanging data has the following main benefits:

- Human readable (useful for debugging, problem resolution).
- Commonly available developer skill.
- Cross platform / technology independent, e.g. 1 service implemented in PHP and running on a Windows Server can exchange messages with a Java application hosted on a Unix server.
- Compact compared to XML, however still much more verbose than other, binary, options described below.
- Especially useful for exchanging data between the back-end services and the Web User Interface as JSON can be natively handled by all browser supporting JavaScript.
- Most familiar format for most web frontend developers.

Disadvantages:

- The main draw-back of JSON is the lack of standards and tool support. Especially defining and validating the contents of a JSON message is limited to humanly readable documentation only, excluding the use of commonly available validators, and thus may suffer different interpretation between the service provider and service client.

Applicability

Should only be used for providing data that will be displayed, directly, in a web browser. I.e. this format should only be used for services invoked from a web browser. Even in this case, whenever data is exchanged in JSON format, both service client and service provider contracts need to be sufficiently flexible to allow dealing with incompatible interpretations of the message format

2.1.3 Comma separated values (CSV)

Description

Data can be exchange as Comma Separated Value filed. This is often useful when data will be generated manually, example as an export from an Excel file.

Advantages:

- Can be exported from MS Excel or another spreadsheet
- May be the most suitable option when interacting with a less technically competent 3rd party.

Disadvantages:

- No standards for defining the message (e.g. various field delimiters are possible)
- Lack of tool support, e.g. validators and transformation

Applicability

Should only be used for exchanging data to be consumed by end users / power users, where the usage will be limited, or no development resources are available.

2.2 Binary formats

Many different options exist. Below some of the most appropriate to EMSA are discussed.

2.2.1 Coherence Portable Object Format (POF)

Description

Coherence has a proprietary data format which is designed to be compact and minimise resource usage.

Advantages:

- Small message size, i.e. very suitable for exchanging messages at high rate over the network
- Limited resource usage, e.g. CPU

Disadvantages:

- Proprietary format which requires the use of Oracle / Tangosol libraries and associated licensing agreements
- Limited language support (only Java, .Net and C++)
- Not humanly readable
- POF requires the developer to implement routines in order to serialize and deserialize the objects. The (de)serialisation code will need to be provided as a library to all service clients.

Applicability

Should only be used in conjunction with Oracle Coherence.

2.2.2 AVRO

Description

Advantages:

- Small message size, i.e. very suitable for exchanging messages at high rate over the network
- Limited resource usage, e.g. CPU

- Navigating an object tree can be easier than equivalent XML
- Message format is well defined

Disadvantages:

- Future / backward compatibility can be trickier compared to XML
- Not humanly readable
- No tools for transforming between message formats or versions

Applicability

Because AVRO is the most often used format in conjunction with Kafka queues, which are being adopted in EMSA by the STAR Streaming project, the preferred binary data exchange format at EMSA will be AVRO. As such, schemas for Vessel Positions and Vessel Voyage Information have been implemented and should be used for the exchange of these types of data between EMSA maritime applications.

2.2.3 Protocol Buffers

Description

This was originally developed at Google and has since been open sourced. As compared to POF it has similar goals for minimising resource usage (CPU and network) whilst also providing support for a larger number of languages. Additionally, usage does not require any licensing.

Advantages:

- Small message size, i.e. very suitable for exchanging messages at high rate over the network
- Limited resource usage, e.g. CPU
- Navigating an object tree can be easier than equivalent XML
- Message format is well defined

Disadvantages:

- Future / backward compatibility can be trickier compared to XML
- Not humanly readable
- No tools for transforming between message formats or versions

Applicability

Not recommended at EMSA, AVRO should be used instead.

2.2.4 Thrift

Description

Is comparable to Protocol Buffers and AVRO. Like AVRO, this format is also maintained by the Apache foundation.

The goals and advantages are very similar, we therefore recommend using only AVRO at EMSA as this will improve interoperability and limit proliferation of similar but different technologies.

Applicability

Not recommended at EMSA, AVRO should be used instead.

2.2.5 Java serialized objects

Description

Java Serialization is an out-of-the-box java feature. It has better performance characteristics than the human readable formats and due to ease of use can be a convenient choice for (short term) storage of data inside of an application.

Advantages:

- Convenience, out-of-the-box java feature

Disadvantages:

- No support for exchanging data with other programming languages
- Dependent on the version and implementation of the JDK.
- Not possible to maintain backwards compatibility other than through maintaining a separate implementation for older versions.

Applicability

Not recommended for exchange of data between applications / services. This format can however be used for transient storage within a single application.

Appendix C Application Monitoring Format

All applications must provide a minimum of one monitoring end-point.

The monitoring end-point is a dedicated and public URL that, when requested, returns an indication of the application health.

The monitoring end-point must be compliant with the following specification:

I. Health Check URL format: https://<FQDN>/healthcheck
II. The /healthcheck request implementation must go through all layers of the application (presentation layer, business layer, database) in order to provide an accurate information about end-to-end status of the application.
III. Making a request to the health check URL, will result in an XML or JSON[5] response with one of the following cases:
    a. if the application is healthy (no issues detected), will return status OK,
    b. if the application is NOT available, it will timeout,
    c. if any problem detected in the application, will return, status NOT_OK, as a minimum. Other information to better diagnose the detected issue could be added.
IV. Others health check URLs can be available for better and deeper monitoring points. In that case, the URL format must be: https://<FQDN>/healthcheck/<monitoring point>

[5] choose format one and maintain consistency across the application

European Maritime Safety Agency
Praça Europa 4
1249-206 Lisbon, Portugal
Tel +351 21 1209 200
Fax +351 21 1209 210
emsa.europa.eu